ROBOGYVER: AUTONOMOUS TOOL MACGYVERING FOR INVENTIVE PROBLEM SOLVING

A Dissertation
Presented to
The Academic Faculty

By

Lakshmi Velayudhan Nair

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering
Department of Robotics

Georgia Institute of Technology

December 2020

ROBOGYVER: AUTONOMOUS TOOL MACGYVERING FOR INVENTIVE PROBLEM SOLVING

Thesis committee:

Dr. Sonia Chernova, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Mark Riedl
School of Interactive Computing
Georgia Institute of Technology

Dr. Devi Parikh
School of Interactive Computing
Georgia Institute of Technology

Dr. Christopher Atkeson
Robotics Institute
Carnegie Mellon University

Dr. Brian Magerko
School of Literature, Media, and Communication
Georgia Institute of Technology

Date approved: September 25, 2020

You can’t use up creativity. The more you use, the more you have.

Maya Angelou

ACKNOWLEDGMENTS

This thesis would not be possible without the wonderful people who guided me on this journey. I would like to start by thanking my amazing advisor, Sonia Chernova, who offered guidance at a time that my future seemed bleak to me. You helped me through some of the roughest patches of my Ph.D., supporting me to actively pursue my research interests. Thank you for being there when I needed the guidance, both in life and research. I would also like to extend my heartfelt gratitude to my committee member Chris Atkeson, who mentored me during my Ph.D. with great career and life advice, and instilled the confidence in me to pursue my academic and professional interests. Thank you for believing in me.

I would also like to thank my amazing committee members: Devi Parikh, Brian Magerko, and Mark Riedl. Without your guidance, this thesis would be incomplete. I am truly honored to have you on my committee, and I am really grateful for all your invaluable feedback and support, that made this thesis possible and also fulfilling.

Thank you to my cohort: Kalesha Bullard, David Kent, Harish Ravichandar, Sid Banerjee, Angel Daruna, Asif Rana, Weiyu Liu, Oviya Thanigaivelan, Evana Gizzi, Nithin Shrivatsav, Devleena Das, Zackory Erickson, Jon Balloch, Reza Ahmedzadeh, Vivian Chu, and Tesca Fitzgerald. Thank you for making my time at GT very memorable. I will miss our lab gaming sessions that pushed me through stressful times. Thank you for listening to my numerous presentations, and for providing insights throughout the course of my thesis.

Lastly, I would like to thank my family. My beloved dad and mom, my caring brothers and sisters: Sujith, Ranjith, Divya, Arya, and nephew Vaishnav, for being my pillars of support. This journey could not even begin without your love and care. You always encouraged me to believe in myself. I also want to thank my fiancé and best friend, Vivek Vijayan, who stuck with me through some of the most daunting times of my life, and ensured that I never gave up. You give me so much strength. I could not possibly imagine a better person by my side, and I look forward to battling whatever the next challenge may be together.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Robots that are situated in the real world are often faced with unforeseen situations that require them to adapt and improvise to be more useful. Particularly in the context of using tools, there may be situations where a robot does not have access to the tools it needs for completing a task. While humans show remarkable improvisation capabilities, similar skills are beyond the scope of robots today. In order to address these scenarios, a resourceful robot should be able to inventively use whatever objects are available to it, in order to replace the missing tool. We refer to this process as "tool macgyvering". Tool macgyvering can be achieved by either substituting the missing tool with an object (tool substitution), or constructing a replacement tool by combining multiple objects (tool construction).

This thesis examines the problem of tool macgyvering, to enable a robot to effectively use available objects to make up for missing tools that are necessary for completing a task. As evidenced by existing research in psychology, tool macgyvering requires reasoning about the task, as well as the physical attributes of the available objects. This thesis seeks to investigate the following hypothesis: **Given a task where the required tools are unavailable, a robot can efficiently perform tool macgyvering by reasoning about the task, and properties of the available objects, including their shape and material**. To support this claim, this thesis contributes: (1) a *formalization of three levels of tool macgyvering* that highlights the levels of complexity involved in tool macgyvering problems; (2) novel algorithms for *tool construction through shape, material and attachment reasoning*, where attachment refers to the different ways in which objects can be combined; (3) a novel algorithm for *tool substitution using shape and material reasoning*; (4) a novel framework that performs *tool macgyvering through arbitration of substitution and construction*, to enable a robot to effectively decide the better of the two solutions for completing the task; and (5) a novel algorithm to perform *tool macgyvering in task planning*, to enable a robot to leverage existing planning algorithms to perform tool macgyvering efficiently.

CHAPTER 1
INTRODUCTION

1.1 Motivation

"*MacGyver, v. - To construct, fix, or modify (something) in an improvised or inventive way, typically by making use of whatever items are at hand; to adapt expediently or ingeniously.*"

- Oxford English Dictionary [1]

A transformative change for robotics is to enable robots to efficiently adapt to handle unforeseen situations. Particularly in the context of tool use, robots may often encounter scenarios where the right tools for completing a task are unavailable to them. In similar situations, humans can remarkably improvise, or *macgyver*[1], with whatever is available. For example, in the failed Apollo 13 moon mission, square CO_2 filters from the Command Module had to be adapted to replace circular filters in the Lunar Module, which was being used as a lifeboat to save power in the Command Module for re-entry. Mission Control, using a set of duplicate parts on the ground, came up with an adapted filter that used objects like a cardboard sheet from a flight manual, tubing from space suits, socks, a towel, plastic bags used to store space suit underwear, and duct tape (See Figure 1.1, left). The adapted filter enabled the astronauts to safely return home [3]. More recently, makeshift ventilators constructed from low-cost 3D printed parts and off-the-shelf materials, such as manual resuscitator PVC bags and motors, have been used to mitigate the widespread equipment shortages during COVID-19 [4, 5, 6] (See Figure 1.1, middle). In simpler examples of tool improvisation, humans often find creative replacements for missing tools, such as using a baby wipes box as a replacement for a mouse pad (See Figure 1.1, right). However, similar improvisation capabilities are currently beyond the scope of existing robotic systems.

[1]The term "macgyver" originates from a popular television series "MacGyver" (1985-1992) [2]

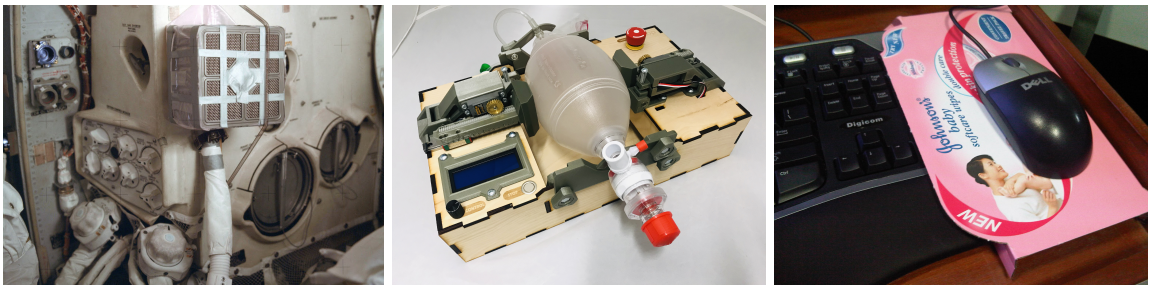


Figure 1.1: Real-world examples of tool improvisation by humans. **Left:** Makeshift CO_2 filter constructed on the Apollo 13 [3]. **Middle:** Makeshift ventilator constructed from low-cost off-the-shelf parts (ApolloBVM) [6]. **Right:** A baby wipes box used to substitute a mouse pad [8].

The goal of this dissertation is to enable robots to macgyver tools in order to solve unexpected problems. Tools are commonly defined as objects that extend the physical influence of the agent [7], and the ability to improvise appropriate tools from available resources greatly increases robot adaptability, enabling robots to efficiently handle failures and uncertainties. These capabilities will be especially useful for robots that explore, as well as work in space, underwater, remote locations on land, disaster sites, and other locations where all possible tools are not easily available.

1.2 Dissertation Overview

This dissertation focuses on the problem of *Tool Macgyvering*, which we define as follows:

Definition 1 *Tool macgyvering is defined as the process of solving tasks where required tools are missing, by either directly substituting the missing tool with a different object, or by constructing a replacement tool from objects available in the environment.*

Thus, tool macgyvering consists of tool substitution (e.g., a mug is used as a substitute tool for a hammer), and tool construction (e.g., a hammer is constructed from a stone and a rod). This dissertation contributes novel algorithms for both tool substitution and tool construction, in order to enable robots to efficiently macgyver tools using available objects. While robots today are mostly confined to the use of predefined tools, there have been prior works focusing on the problem of tool substitution for improving robot adaptability [9, 10,

11, 12, 13]. However, these approaches do not explicitly reason about material properties of objects when selecting substitute tools, and this research contributes an approach for improving on prior tool substitution work by incorporating material reasoning. In contrast to the problem of tool substitution, however, there has been very limited work focusing on enabling robots to construct the required tools from available objects. As such, novel algorithms for tool construction forms one of the core contributions of this dissertation, with our work being one of the first to demonstrate tool construction on a physical robot.

The approaches presented in this dissertation are inspired and motivated by prior research in psychology. In particular, prior work that has focused on tool-making in children, has demonstrated the importance of reasoning about the task, as well as the shape and material of available objects in order to effectively make tools [14, 15, 16, 17]. For instance, in the context of a "hooking task" that requires bending a pipe-cleaner into a hook to retrieve an object, children had to reason about the task, the material properties of the pipe-cleaner, and its required shape in order to successfully complete the task. This was shown to be a challenging cognitive problem, emerging much later in the development process of children (around 8 years of age), due to the ill-structured nature of the task in which components of the solution must be both retrieved and coordinated [17]. Specifically with respect to object properties, prior work has also theorized that humans use a weighted combination of modalities to efficiently determine object properties [18], and that reasoning about shape and material properties enable humans to accurately assess the stability of an object [19]. Inspired by this body of work, we present our thesis statement below.

1.2.1 Thesis Statement

Given a task where the required tools are unavailable, a robot can efficiently perform tool macgyvering by reasoning about the task, and properties of the available objects including their shape and material.

1.2.2 Contributions

To support this claim, this dissertation makes the following contributions:

- **Formalization of Three Levels of Tool Macgyvering:** (Chapter 3) The first contribution of this dissertation seeks to formalize the complexity of tool macgyvering problems. In the simplest case, a prototypical tool or action for solving a task is known a-priori, and a substitute or constructed tool is used to accomplish it, e.g., using a stone in place of a hammer to join two pieces of wood by hammering a nail. In more complex problems, the task is reformulated such that an alternate action can be applied for completing the task, e.g., gluing the two pieces of wood as opposed to hammering nails. Intuitively, there are several creative solutions for completing a task that lie on a spectrum of difficulty in terms of the reasoning or cognitive capabilities involved. This dissertation introduces a formalization of three levels highlighting the varying degrees of complexity involved in tool macgyvering problems [20].

- **Tool Construction Using Shape, Material and Attachment Reasoning:** (Chapter 4) This dissertation contributes a novel tool construction algorithm that utilizes a multi-objective function to identify, and rank objects that are viable candidates for constructing an effective replacement tool. The multi-objective function accounts for physical attributes of the available objects and involves three objectives, each of which is learned through supervised learning techniques. The three objectives concern the *shape* of the objects, the *material* properties of the objects, and the *attachment capabilities* as to whether the objects can be attached together to construct a tool. We implement our approach on a 7-DOF robotic arm, and demonstrate the ability of our algorithm to efficiently reason about a wide range of objects through autonomous construction of six different tools [20, 21, 22].

- **Tool Substitution Using Shape and Material Reasoning:** (Chapter 5) This dissertation contributes a novel tool substitution algorithm that jointly reasons about the

shape and material of available objects, unlike prior work in the area that only reasons about shape to identify suitable substitutes [23]. In our experiments, we validate our approach on a set of real objects for the substitution of six different tools. Our results demonstrate that combining material reasoning with shape, helps significantly improve the performance of tool substitution when compared to existing approaches.

- **Tool Macgyvering Through Arbitration of Substitution and Construction:** (Chapter 6) Arbitration is the process of selecting one among several given behaviors [24]. This dissertation contributes a novel framework for performing tool macgyvering by integrating tool substitution and construction through intelligent arbitration. We present three arbitration techniques that enable a robot to decide between tool substitution and tool construction as the more appropriate solution, given a set of objects. We evaluate our approaches in a set of real-world experiments, and present the relative strengths and weaknesses of the three approaches. Our results demonstrate that the proposed approaches are able to effectively select between tool substitution and construction in terms of concurring with the choices made by a human [22].

- **Tool Macgyvering in Task Planning:** (Chapter 7) This dissertation contributes the *Feature Guided Search* algorithm, that integrates tool macgyvering with existing planning approaches to enable the robot to efficiently perform tool construction in the context of task planning. We demonstrate that our approach significantly improves the computational performance of standard heuristic search algorithms, such as A^*, when applied to the problem of tool construction. Additionally, our results demonstrate that our approach is able to flexibly adapt task plans based on the available objects, to select the appropriate action for using the constructed tool [25]. This dissertation is the first work to demonstrate the application of heuristic search algorithms to the complex combinatorial problem of tool construction.

1.2.3 Outline of Dissertation Document

This dissertation is organized as follows. Chapter 2 discusses the technical background and related work for the research questions examined in this dissertation. Note that each chapter (beginning with Chapter 3) opens with the key research question that guides the work in that chapter. The formalization of the complexity levels of tool macgyvering problems is presented in Chapter 3, along with a discussion of the research questions in the context of the presented formalization. Chapter 4 discusses the algorithmic contributions for tool construction, and Chapter 5 focuses on the algorithmic contributions for tool substitution. Chapter 6 presents a tool macgyvering framework that combines tool substitution and tool construction through arbitration. Chapter 7 introduces novel algorithms for performing tool construction in the context of task planning. Finally, we conclude and discuss interesting open questions in Chapter 8. The Appendix discusses additional implementation details, as well as information about the code and datasets developed in this dissertation.

CHAPTER 2
PROBLEM FORMULATION AND RELATED WORK

In this chapter, we present our problem formulation that represents a mathematical description of our thesis statement. We begin by discussing preliminaries with respect to planning problems in AI in subsection 2.1.1, and present relevant notations and terms that are necessary prerequisites. In subsection 2.1.2, we present the problem formulation that guides this dissertation, followed by a discussion of the relevant literature in section 2.2, as it relates to our problem formulation. We summarize with a table of notations in section 2.3.

2.1 Background

The goal of this dissertation is to enable robots to efficiently macgyver tools from available objects when the required tools for completing a task are missing. Hence, as input the robot is provided with a set of objects in its environment, and a task that requires certain tools that are unavailable. The robot must then generate a solution for completing the task, that involves either substituting or constructing the missing tools using the available objects. In order to perform tool substitution, our approach reasons about the shape and material properties of the objects to identify good substitutes for the candidate tool. For tool construction, the robot also reasons about the different ways in which objects can be attached to construct the required tool, in addition to the individual shape and material properties of the objects. As a note, this document uses the phrases "candidate objects" and "candidate parts" interchangeably. Both phrases refer to the objects that are potential candidates for substituting or constructing tools. In the following chapters, there are instances where we denote tool construction by *cons*, and tool substitution by *subs*.

2.1.1 Planning Problem

Prior to presenting our problem statement, we first define a planning problem in AI. Planning in AI seeks to enable an artificial agent to synthesize a sequence of actions, i.e., a plan, to achieve its goals [26]. A classical formulation of a planning problem defines three inputs: (1) a description of the initial state of the world in some formal language, (2) a description of the agent's goal (that is, what behavior is desired) in some formal language, and (3) a description of the possible actions that can be performed (again, in some formal language) [26]. In this dissertation, we use the Planning Domain Definition Language (PDDL) to generate the three descriptions [27]. More formally, let S denote the set of states, A denote the set of actions, γ denote state transitions, s_i denote the initial state, and s_g denote the goal state. Then, a planning problem consists of a domain definition $\mathcal{P}_\mathcal{D} = (S, A, \gamma)$, and a problem or task definition $\mathcal{P}_\mathcal{T} = (\mathcal{P}_\mathcal{D}, s_i, s_g)$ [28]. Note that, in the remainder of this dissertation, we always assume that the robot is missing some required tools for completing the provided task, even if it is not explicitly stated. In addition to the planning problem and domain definitions, we also assume that the robot is provided with a set of n candidate objects represented by their point clouds. We denote the set of objects as O.

2.1.2 Problem Formulation

Given a set of n candidate object point clouds $O = \{o_1, o_2, ...o_n\}$, and a planning task that is defined by a domain description $\mathcal{P}_\mathcal{D}$ and a problem description $\mathcal{P}_\mathcal{T}$, how can we enable the robot to perform tool macyvering for accomplishing the task goal s_g?

Our overarching approach for addressing this problem involves multi-objective optimization through scalarization [29]. A multi-objective optimization problem deals with more than one objective or criteria that a potential solution must optimize for [30]. Scalarization is a method for optimizing multi-objective functions by combining the different objective functions into a single solution through a weighted summation [29]. We compute a multi-objective function, denoted by Φ, that indicates the fitness of the objects in O as

candidates for substituting or constructing for the missing tool that is required to accomplish the task. For tool substitution, the objective function evaluates the shape and material properties of the candidate objects. For tool construction, the objective function evaluates the attachment capabilities of the candidate objects, in addition to their shape and material properties. Further, we use supervised learning techniques to compute each objective within the multi-objective function.

In each of the following chapters (Chapters 4-6), this dissertation contributes novel algorithms that seek to address specific parts of the problem formulation, with Chapter 7 finally converging to address the full problem formulation by incorporating the algorithms developed in the previous chapters.

2.2 Related Work

In the following sections, we present existing research that is relevant to the questions examined in this thesis. In subsection 2.2.1, we discuss existing research in the broader space of creative problem solving or macgyvering. We then discuss relevant research specific to tool construction (subsection 2.2.2) and tool substitution (subsection 2.2.3). Since our thesis statement involves reasoning about the shape and material properties of objects, we present research that has focused on point cloud representations for shape reasoning (subsection 2.2.4), and object representations for material reasoning (subsection 2.2.5). In subsection 2.2.6, we present research that is relevant to arbitration of different behaviors, and finally conclude with ethological studies of tool macgyvering in animals (subsection 2.2.7).

2.2.1 General Approaches to Creative Problem Solving

Prior work by Sarathy and Scheutz have focused on formalizing creative problem solving in the context of planning problems [31, 28]. They define the notion of "Macgyver-esque" creativity as embodied agents that can "*generate, execute, and learn strategies for identifying and solving seemingly unsolvable real-world problems*" [31]. They formalize Mac-

gyvering problems (MGP) with respect to an agent t, as a planning problem in the agent's world $\mathbb{W}_t$, that has a goal state g currently unreachable by the agent. As described in their work, solving an MGP requires a domain extension or contraction through perceiving the agent's environment and self. Additionally, Gizzi et al. have formalized the definition of creative problem solving in terms of problems that lie at the intersection of AI and Computational Creativity, requiring the modification of the initial conceptual space of an agent [32]. Prior work by Sarathy also provides an in-depth discussion of the cognitive processes involved in creative problem solving in detail, by leveraging existing work in Neuroscience [33]. In similar work, Olteţeanu and Falomir have looked at the problem of Object Replacement and Object Composition (OROC) situated within a cognitive framework called, the Creative Cognitive Framework (CreaCogs) [34]. Object replacement and Object Composition are analogous to tool substitution and tool construction, respectively. Their work utilizes semantic relationships between objects in order to reason about alternate uses for objects to creatively solve problems, e.g., using a matchbox as a candle-holder. However, the semantic relationships themselves are encoded a-priori. Similar work by Freedman et al. has focused on the integration of analogical reasoning and automated planning for creative problem solving, by leveraging semantic relationships between objects [35]. They present the Creative Problem Solver (CPS), that uses large-scale knowledge bases to reason about alternate uses of objects for creative problem solving. In contrast to reasoning about objects, prior work by Gizzi et al. has looked at the problem of discovering new actions for creative problem solving, enabling the robot to identify previously unknown actions [36]. Their work applies action segmentation and change-point detection to previously known actions in order to enable a robot to discover new actions. The authors then apply breadth-first search and depth-first search in order to derive planning solutions using the newly discovered actions. Similar work by Suárez-Hernández et al. has focused on the autonomous synthesis of action schemas in PDDL, from a set of execution traces, by optimizing for a cost function that incorporates the number of adding effects, deleting effects

and predicates in preconditions of actions [37].

In contrast to these approaches, this dissertation focuses on the specific problem of tool macgyvering, by reasoning about the shape and material properties of objects observed by the robot's sensors, as opposed to semantically or symbolically encoded a-priori knowledge. Further, our research introduces a formalization that captures the varying degrees of complexity involved in tool macgyvering problems.

2.2.2 Tool Construction

Existing research in robotics has primarily focused on tool use [38, 39, 40, 41], with little prior work in tool construction. Some recent work by Erdogan and Stilman has focused on the Automated Design of Functional Structures (ADFS), involving construction of navigational structures, e.g., stairs or bridges [42]. They introduce a framework for effectively partitioning the solution space by inducing constraints on the design of the structures. Further, Tosun et al. have looked at planning for construction of functional structures by modular robots, focusing on identifying features that enable environment modification in order to make the terrain traversable [43]. In similar work, Saboia et al. have looked at modification of unstructured environments using objects, to create ramps that enhance navigability of the terrain [44]. More recently, Choi et al. extended the cognitive architecture ICARUS to support the creation and use of functional structures such as ramps, in abstract planning scenarios [45]. Their work focuses on using symbolically encoded attributes of objects that are specified a-priori, such as weight and size, in order to reason about the construction and stability of navigational structures. More broadly, these approaches are primarily focused on improving robot navigation through environment modification as opposed to the construction of tools.

Some existing research has also explored the construction of simple machines such as levers [46, 47]. Erdogan and Stilman have formulated the construction of simple machines as a constraint satisfaction problem wherein the constraints represent the relationships be-

tween the design components [48]. The constraints in their work limit the variability of the simple machines that can be constructed, focusing only on the placement of components relative to one another, e.g., placing a plank over a stone block to create a lever. Similar work by Levihn and Stilman has looked at the use of environmental objects as tools in the context of a door opening task [47]. They introduce an approach that efficiently explores the large and intractable space of object interactions by back-propagating physical constraints between useful combinations of objects.

Apart from the use of environmental objects as tools, Wicaksono and Raymond have focused on using 3D printing to fabricate tools from polymers [49]. Their work encodes the geometries of specific sub-parts of tools, and enables the robot to experiment with different configurations of the sub-parts when fabricating tools. The robot further evaluates the success of the fabricated tool through repeated trials of using the tool for performing the task. Other work by Wang et al. [50], have focused on fabrication of novel tools from thermoplastics as opposed to using environmental objects. They assess the energetic costs involved in the tool manufacturing process in order to enable the robot to efficiently craft novel tools using thermoplastics. The robot uses the created tools in order to perform a set of specified tasks to evaluate the effectiveness of the tool.

In contrast to the research described above, this dissertation focuses on the construction of tools from available objects as opposed to tool fabrication from polymers. Furthermore, in relation to existing research in tool construction, our thesis contributions reason about a wide range of objects, and multiple modes by which the objects may be attached, in order to construct tools. As such, this dissertation is the first work to demonstrate tool construction on a physical robot. Additionally, our thesis also performs tool construction in the context of task planning, being one of the first to investigate the application of heuristic search algorithms to the problem of tool construction.

2.2.3 Tool Substitution

In tool substitution, prior work by Boteanu et al. has explored the use of large-scale semantic networks for reasoning about tool substitutes [12]. They introduce a framework that enables robots to reason about task failures due to missing objects to perform plan repair. The task plans are encoded in the form of Hierarchical Task Networks (HTNs). In the case of missing objects, the robot identifies and scores the fitness of candidate substitutes based on the context of the specified HTN, and based on the similarities between the missing and the candidate object as encoded in the semantic network. In contrast to semantic reasoning, some existing work has looked at leveraging visual similarities between tools, to identify good substitutes [10, 51]. Specifically, visual reasoning compares visual properties of the missing object and the candidate objects in order to assign a fitness score that is then used to select appropriate tool substitutes. Abelha et al. introduced an approach that computed the fitness score through the use of non-linear optimization techniques such as Levenberg-Marquardt, applied to objective functions that represent the geometric properties of objects [10]. More closely related to our approach, Schoeler and Wörgötter use supervised learning with neural networks in order to compute the fitness of the candidate objects [9]. The neural networks learn function-to-shape correspondence of objects, and identify substitutes for a given tool using part-based shape matching.

In contrast to the approaches described above, this dissertation introduces a tool substitution algorithm that jointly reasons about the shape *and material* of candidate objects to identify substitute tools. This enables the robot to deal with situations where appropriately shaped objects, may not be made of suitable materials, e.g., a clay block cannot be used for hammering. Additionally, we use supervised learning with dual neural networks[1] in order to learn both function-to-shape and function-to-material correspondences of objects.

[1] Also referred to as Siamese Neural Networks. However, we refrain from using the term "Siamese".

2.2.4 Tool Representations for Shape Reasoning

Existing work in robot tool use has introduced several descriptors for representing 3D tool point clouds [52, 53, 10, 9]. Descriptors are n-dimensional vectors of real values describing the shape of the point clouds. These descriptors can be classified into parametric and non-parametric shape descriptors. For parametric descriptors, each value indicates a specific geometric aspect of the objects that it represents, e.g., height of a cylinder, or the radius of a sphere. Thus, parametric descriptors are often used to represent geometrically well-defined shapes. In contrast, the values of a non-parametric shape descriptor does not refer to any meaningful physical attribute of the objects, and can be used to effectively represent irregularly shaped point clouds.

Among parametric descriptors, prior work has explored the use of Superquadrics for tool representation [10, 20]. Superquadrics (SQs) refer to the family of geometric shapes that includes quadrics, but allows for arbitrary powers instead of just power of two [54]. SQs are represented by 13 parameters: 3 for scale in each dimension, 2 for shape variance, 3 for Euler angles, 2 for tapering parameters and 3 for the central point/mean. Thus, each parameter holds a specific geometric meaning.

The most widely used of non-parametric shape descriptors include, Ensemble of Shape Functions (ESF) [52] and Signature of Histogram of Orientations at Centroid (SHOTC) [53, 51]. ESF is a shape descriptor consisting of 10, 64-bin sized histograms (640-D vector) describing the shape properties of a point cloud. SHOTC is a descriptor that encodes the information of a surface enclosed within a spherical supporting structure. The sphere is centered at the centroid of the point cloud for a radius equal to the max dimension of the point cloud, and divided into bins of volumes [9]. Local histograms representing the shape properties are computed within each bin, resulting in a 352-D vector.

In this dissertation, we investigate the use of both parametric shape descriptors using SQs, and non-parametric shape descriptors using ESF and SHOTC, for representing the input point clouds.

2.2.5 Tool Representations for Material Reasoning

Material properties play an important role when detecting the appropriateness of parts for tool construction, e.g., when constructing a hammer, wooden or metallic parts are preferred over foam. Perlow et al. describe an approach for detecting appropriate raw materials for object construction, and demonstrate their work in the simulated world of Minecraft [55]. Their work uses dual neural networks to predict the material similarity of two objects based on input images of the objects. Several other vision-based approaches to material recognition have been previously explored [56, 57, 58]. These approaches are often sensitive to lighting conditions and viewing angle. Additionally, objects can be deceptive in appearance, resulting in mis-identification, e.g., a foam block with wooden texture would be wrongly detected as wood. In contrast to visual reasoning, spectral reasoning uses a spectrometer to scan and measure the reflected intensities of different wavelengths, in order to profile and reason about material properties of objects. Prior work in the use of spectral reasoning for determining material class of objects has shown highly promising results, with an overall material class prediction accuracy of 94.6% [59].

In this thesis, we investigate the use of spectral reasoning with a handheld spectrometer in order to determine material properties of objects, and to evaluate their fitness for macgyvering tools. Specifically, our novel contribution includes the use of spectral readings as inputs into dual neural networks in order to predict material fitness of candidate objects.

2.2.6 Arbitration of Behaviors

While there has not been prior work specifically focusing on the arbitration of tool substitution and tool construction, behavior-based design methodologies often explore coordination mechanisms for different robot behaviors [60, 61, 24]. Given a set of behaviors, the goal of the coordination or arbitration mechanism is to generate an output behavior that is either one, or a combination of the input behaviors. Two arbitration strategies have been commonly explored to accomplish this, namely, Action Selection and Behavioral Fusion

[24]. In action selection, each behavior is associated with a value function, denoted by Ψ, that dictates the behavior chosen at any given instant. Thus, only one of the input behaviors is selected. In contrast, behavioral fusion generates a weighted summation of the input behaviors, often used in navigational tasks.

Given the nature of our problem in this work, we use action selection to arbitrate between tool substitution and construction. We contribute three value functions based on the multi-objective functions developed in this dissertation, in order to select between tool substitution or tool construction as the more appropriate solution for solving the task.

2.2.7 Tool Macgyvering by Animals

Particularly among biological sciences, tool construction has been extensively studied in communities of prehistoric humans [62, 63], mammals [64, 65], and birds [66, 7], in order to explore the emergence of more sophisticated levels of intelligence. Beck et al. [17], identify two classes of problems in tool-making: *tool innovation* and *tool manufacturing*. Tool manufacturing refers to a class of problems where the required tool for solving the task is known from prior experience, and the tool must be constructed from available objects. In contrast, tool innovation requires imagining the ideal tool required for the given task, followed by its construction from available objects.

The tool construction work introduced in this dissertation is inspired by the problem of tool manufacturing, where the robot is provided a-priori knowledge regarding the required tool, and must construct an appropriate replacement for it using the available objects.

2.3 Notations Used

Table 2.1 summarizes all notations introduced in this chapter. Though each chapter that follows is self-contained and will define relevant notation, to the extent possible, the notation below remains consistent throughout the dissertation document.

Table 2.1: Table of Notation

O	set of candidate objects
n	number of candidate objects
S	set of world states
A	set of actions that the robot can perform
$\mathcal{P}_{\mathcal{D}}$	domain definition of planning task
$\mathcal{P}_{\mathcal{T}}$	problem definition of planning task
Φ	multi-objective function
Ψ	value function for arbitration
ϕ	single objective function
cons	tool construction
subs	tool substitution

CHAPTER 3
COMPLEXITY OF TOOL MACGYVERING PROBLEMS

How can we formalize the complexity of tool macgyvering problems?

In this chapter, we propose that reasoning about the notion of affordances can help formalize the complexity of tool macgyvering problems. We begin by outlining the need for such a formalization in tool macgyvering. We then discuss the definition of affordances, and prerequisites that are relevant to our formalization. We then present the first contribution of this dissertation in terms of formalizing three levels of complexity involved in tool macgyvering problems. Lastly, we discuss the research questions examined in our dissertation in terms of the presented formalization.

3.1 Need for Formalizing the Complexity of Tool Macgyvering Problems

Prior work by Sarathy and Scheutz have focused on formalizing macgyvering in the context of planning problems [31, 28]. They formalize macgyvering problems (MGP) with respect to an agent t, as a planning problem in the agent's world $\mathbb{W}_t$, that has a goal state g currently unreachable by the agent. As described in their work, solving an MGP requires a domain extension or contraction through perceiving the agent's environment and self. However, their work does not characterize the spectrum of complexity involved in tool macgyvering. For instance, consider using a stone in place of a hammer to join two planks by hammering a nail. In the absence of a stone or an alternate object, the task may be reformulated such that an alternate action can be applied for completing the task, e.g., gluing the two pieces of wood as opposed to hammering nails. Intuitively, we see that in each case the robot relaxes assumptions associated with the tools and actions that can be applied for solving the task in order to come up with creative solutions. The first contribution of this thesis seeks to formalize this intuition by introducing three levels of macgyvering that capture the

different complexity levels. There are two key benefits associated with this formalization:

- **Enable the creation of a taxonomy of tool macgyvering problems:** A taxonomy is defined as "*a method of classifying a vocabulary of terms for a specific topic according to specific laws or principles*" [67]. The formalization of the complexity levels will allow structured development and classification of existing, as well as future algorithms that are developed for tool macgyvering.

- **Enable the robot to autonomously assess the complexity of a task:** When provided with a task, the complexity formalization will enable the robot to assess the difficulty of the task that it is faced with, and to generate appropriate solutions to the problem based on the difficulty level.

In the following sections, we discuss how the notion of affordances allows us to formalize the complexity of tool macgyvering. We discuss how tool substitution and tool construction are all related to the representation of tool affordances. Below, we first define and characterize tool affordances, and then show that the complexity levels of tool macgyvering problems can be formalized based on the extent to which the tool affordance representation must be adapted.

3.2 Affordances

The term "affordance" was first introduced by psychologist J.J. Gibson (1977) [68]. We use the ecological definition of "action possibilities" that appear between an agent and the environment, which is commonly used in robotics [69, 70]. For example, if a robot performs the action of tilting a cup to pour water from it, the cup is said to have the affordance "pour-able". Computationally, affordances are defined as unique relationships between objects (O), the actions those objects can be used for (A), and the effects (E) of applying the actions with the objects [71]. Problems that are solved using tool construction and substitution are defined by their two primary constituents: a *task goal* G and a *solution*

affordance S. We define goals as persistent relationships between objects, such as "*isAttached*(painting, wall)" to denote the task of attaching a painting to the wall. We use the tuple $S = (O, A, E)$ to denote an affordance solution for accomplishing the goal G.

3.3 Affordance Equivalences

Prior work on affordance representations by Andries et al., define the notion of *affordance equivalences* as affordances that generate equivalent effects [72]. In particular, they introduce three types of affordance equivalences, namely object equivalence, action equivalence, and object-action equivalence. For our formalization, we focus specifically on object equivalence, and object-action equivalence. Object equivalence (denoted by O^{eq}) occurs when the same action applied to two different objects generates an equivalent effect. Similarly, object-action equivalence (OA^{eq}) occurs when different actions applied to different objects result in equivalent effects. For the purposes of our formalization, we introduce a third equivalence class, object-action-effect equivalence (OAE^{eq}), as occurring when different actions applied to different objects generate different effects, but that those effects accomplish the same task goal G. Table 3.1 presents an example of all three levels of equivalence for the task goal $G = isAttached(board1, board2)$ and reference solution $S_R = (screwdriver, turn, tighten\_screw)$. Note that, OAE^{eq} occurs when the goal of attaching the boards is achieved, but without using the original tool, action or effect, such as by tying the boards together with a rope instead of tightening the screw. Below, we show that tool macgyvering problems can be broken down into three classes, based on which elements of the reference solution S_R are assumed to remain unchanged.

3.4 The Three Levels of Tool Macgyvering

This section presents the first contribution of this dissertation. We define three levels of macgyvering based on the equivalence classes described in the previous section, and show how the relative complexity of tool-based problems can be assessed directly from the affor-

Table 3.1: Levels of Macgyvering for the task of attaching two wooden boards. The reference solution (S_R) is to *turn* a screw with a *screwdriver* to *tighten* it. The affordance equivalences (O^{eq}, OA^{eq} and OAE^{eq}) are indicated for each level for both tool substitution (S) and construction (C).

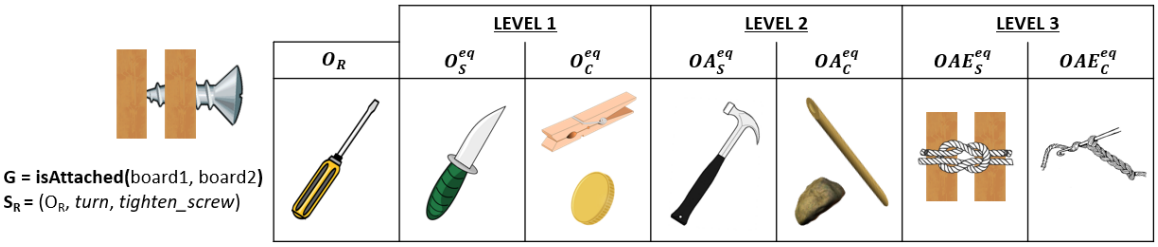


Table 3.2: The three Levels of Macgyvering; "—" indicates the action or effect of the reference tool is being preserved.

	LEVEL 1		LEVEL 2		LEVEL 3	
	O_S^{eq}	O_C^{eq}	OA_S^{eq}	OA_C^{eq}	OAE_S^{eq}	OAE_C^{eq}
OBJECT	O_S	O_C	O_S	O_C	O_S	O_C
ACTION	—	—	A_S	A_C	A_S	A_C
EFFECT	—	—	—	—	E_S	E_C

dance equivalency representation. Each macgyvering level consists of two variants based on whether reference objects are substituted or constructed. *Tool substitution*, denoted by the subscript $_S$, refers to the case in which an existing tool can be used to replace a reference tool. *Tool construction* on the other hand, denoted by the subscript $_C$, refers to the case in which no substitute for the reference tool exists, and a replacement tool must be constructed. The definitions below, also summarized in Table 3.2, are with respect to a task with goal G, and a reference solution defined by $S_R = (O_R, A_R, E_R)$.

3.4.1 Level 1: Object Improvisation

The first level, also called object improvisation, addresses tasks with solutions that can be achieved either through object substitution (O_S^{eq}) or object construction (O_C^{eq}) alone, where the newly found or created object shares the action and effect of the reference solution:

Definition 2 *Object improvisation is defined as the set of tool macgyvering solutions that*

involve a substituted tool O_S or a constructed tool O_C, such that the solution affordance $S = (O_S, A_R, E_R)$ or $S = (O_C, A_R, E_R)$.

As shown in Table 3.1, in the case of O_S^{eq} a knife can be used in place of a screwdriver to *turn* and tighten a screw, and in the case of O_C^{eq}, a clothespin and a coin can be combined to create a tool identical in function to the screwdriver.

3.4.2 Level 2: Action Improvisation

The second level, also called action improvisation, addresses tasks for which no viable object equivalence solution exists, and the goal is achieved by substituting, OA_S^{eq}, or constructing, OA_C^{eq}, a tool with a non-reference action where the object-action pair yields the same effect as the reference solution.

Definition 3 *Action improvisation is defined as the set of tool macgyvering solutions that involve a substituted tool O_S or a constructed tool O_C, such that the solution affordance $S = (O_S, A_S, E_R)$ or $S = (O_C, A_C, E_R)$, where $A_S, A_C \neq A_R$.*

In our example of attaching two boards, in the case of OA_S^{eq}, a hammer can be used to push the screw in with a *hitting* action instead of using the screwdriver with a *turn* action. Both actions accomplish the effect of tightening the screw. In the case of OA_C^{eq}, the same hitting action would be used, but the hammer would need to be constructed, such as by combining a stick and a rock.

3.4.3 Level 3: Sub-goal Improvisation

The third level, also called sub-goal improvisation, addresses tasks for which no viable object-action equivalence solution exists, and the goal G is achieved by substituting, OAE_S^{eq}, or constructing, OAE_C^{eq}, a tool with a non-reference action and non-reference effect.

Definition 4 *Sub-goal improvisation is defined as the set of tool macgyvering solutions that involve a substituted tool O_S or a constructed tool O_C, such that $S = (O_S, A_S, E_S)$ or*

$S = (O_C, A_C, E_C)$, *where* $A_S, A_C \neq A_R$ *and* $E_S, E_C \neq E_R$.

In our running example, in the case of OAE_S^{eq} a rope can be used to *tie* two pieces of wood together, and in the case of OAE_C^{eq} a rope would first be knitted out of strands of yarn and then used to *tie*. Both scenarios no longer accomplish the effect of tightening the screw, but accomplish the task goal of attaching the boards.

As can be seen from the examples, developing a robotic system that can solve the macgyvering problem becomes progressively harder at each level as the robot requires the capability to reason about the task goal, the actions afforded by each object, and their resulting effects, at various levels of abstraction. In each level, the robot relaxes the constraints of the problem from objects to actions to effects in order to derive creative solutions.

3.5 Contributions

This chapter answers the first research question of *"How do we formalize the different levels of complexity in tool macgyvering problems?"*. We leverage the concepts of affordances and affordance equivalences to introduce three levels of complexity in tool macgyvering. The three levels can help guide future research and development of tool macgyvering algorithms, by presenting a taxonomy that organizes the spectrum of complexity for this problem. On the basis of the three levels that we introduced, we now discuss how the different research questions examined in this dissertation relate to the macgyvering levels. Thus, the levels of macgyvering also act as a taxonomy for classifying the algorithms that are developed in this dissertation.

3.5.1 Research Questions in Relation to The Three Macgyvering Levels

This thesis contributes algorithms for solving the first and second levels of macgyvering, namely, level 1: object improvisation, and level 2: action improvisation. An important point to note is that in terms of implementation, level 2 macgyvering encompasses level 1 macgyvering. For instance, once an appropriate alternate action is identified for level 2

macgyvering, the problem reduces to that of level 1 macgyvering. This nesting of levels allows incremental development of algorithms, in that any approaches that are developed for level 1 macgyvering can be extended to perform level 2 macgyvering. Thus, our algorithmic contributions for level 1 is reapplied for level 2 macgyvering, with the additional component that the robot reasons about alternate actions. We first revisit our problem formulation, and then present the research questions in terms of the levels of macgyvering:

Given a set of n candidate object point clouds $O = \{o_1, o_2, ...o_n\}$, and a planning task that is defined by a domain description $\mathcal{P}_\mathcal{D}$ and a problem description $\mathcal{P}_\mathcal{T}$, how can we enable the robot to perform tool macyvering for accomplishing the task goal s_g?

We can decompose the problem formulation into the following sub-problems that relate directly to the contributions of this dissertation:

- **Tool Construction**: *How can we enable the robot to perform tool construction?* In the case of level 1 tool construction, the robot is provided with a reference action, and must construct an appropriate replacement tool for accomplishing the action. To address this problem, we introduce a multi-objective function that reasons about shape, material and attachment capabilities of objects to perform tool construction. We explore this in Chapter 4.

- **Tool Substitution**: *How can we enable the robot to perform tool substitution?* In the case of level 1 tool substitution, the robot is provided with a reference action, and must directly substitute an available object for the missing reference tool. To address this problem, we introduce a multi-objective function that reasons about the shape and material of available objects to perform tool substitution. We explore this research question in detail in Chapter 5.

- **Tool Macgyvering Through Arbitration of Substitution and Construction**: *How can we enable the robot perform object improvisation?* Our contribution here focuses on performing level 1 macgyvering, given a reference action. To address this

problem, we introduce a tool macgyvering framework that integrates tool substitution and construction through intelligent arbitration. To do so, we present three arbitration approaches that enable the robot to decide between tool substitution and tool construction. We explore this in Chapter 6.

- **Tool Macgyvering in Task Planning**: *How can we enable the robot to perform action improvisation?* The final contribution of our dissertation converges on our original problem formulation, and specifically focuses on performing level 2 macgyvering (action improvisation), given the domain and problem definitions for a task. To address this problem, we contribute the Feature Guided Search approach that uses the multi-objective functions developed in the previous chapters to enable the robot to reason about replacement tools *and* alternate actions for successfully completing the task. We explore this research question in Chapter 7.

CHAPTER 4
TOOL CONSTRUCTION

How can we enable robots to perform tool construction?

In this chapter, we demonstrate that reasoning about shape and material properties of objects, and the different ways in which objects can be attached, enables robots to effectively perform tool construction. We first revisit related work, and discuss unaddressed challenges in tool construction. We then present our algorithmic contributions and experimental results. Finally, we summarize our key findings.

4.1 Challenges in Tool Construction

As discussed in the related literature in Chapter 2, existing work in robotics has mainly focused on tool use [38, 39, 40, 41], with little prior work in tool construction. Some of the key unaddressed challenges in tool construction that the contributions of this chapter seeks to tackle are as follows:

- **Shape, material, and attachment reasoning for tool construction:** Most prior work in robotics has focused on the construction of navigational structures such as ramps and bridges, rather than tools [42, 43, 44]. Further, these approaches do not reason about shape or material properties of objects, nor the different ways in which objects can be combined together to construct tools, rather focusing on the placement of one object over another, e.g., placing a plank over a stone block to construct a lever.

- **Evaluating pierceability of objects:** Prior work has looked at classifying material properties of objects, either using visual or spectral data [73, 59, 55]. However, there has been no prior work that has looked at predicting material pierceability from input spectral measurements of the object. Predicting pierceability helps identify whether

two objects can be attached by piercing one with another, e.g., piercing a foam block with a screwdriver.

- **Evaluating tool construction approaches:** Since tool construction is a relatively unexplored problem, there are no existing benchmarks or metrics for standardized evaluation of tool construction approaches. While some prior work has looked at theoretical models for assessing the creativity of a robot through the "Macgyver Test" [31], the authors do not discuss specific tests or benchmarks for conducting the evaluation. This requires proposing datasets and metrics that can support the development and evaluation of tool construction algorithms.

In contrast to prior work, this chapter contributes a novel approach to reason about the shape, material, and attachment capabilities of objects to perform efficient tool construction. We also present a novel approach for predicting the pierceability of objects from input spectral measurements. Further, we develop a dataset for training predictive models (made publicly available[1]), and introduce metrics that are useful for evaluating the performance of tool construction approaches. We present our results on a physical 7-DOF robot arm.

4.2 Tool Construction Approach

Tool construction is a complex combinatorial problem that involves exploring the space of all possible candidate object configurations. Given the set of all n candidate objects as $O = \{o_1, o_2, ..., o_n\}$, the total space of configurations or permutations for the n objects is ${}^nP_2 + ... + {}^nP_n$, assuming at least two and at most n objects must be combined to construct the tool. This results in a prohibitively large problem space that is combinatorial in the number of candidate objects. In this dissertation, we make the simplifying assumption that the number of objects required to construct the tool is equal to the number of components in the desired output tool[2]. That is, there is a one-to-one correspondence between candidate

[1] https://github.com/Lnair1993/Tool_Macgyvering

[2] Most tools have two key components, grasp part and action part [74, 11]

objects and constructed tool components. For instance, for constructing a hammer which has a handle and a head, only two object constructions are explored, one object for the handle and one for the head. For n candidate objects available for tool construction, and m tool components, this reduces the overall combinatorial search space to ${}^nP_m$. We denote the set of all permutations of the m objects as $T = \{T_1, T_2, ...\}$, where $|T| = {}^nP_m$ and $T_i = (o_1, ..., o_m)$ is a tuple representing a specific permutation of m objects.

4.2.1 Overview of Approach

In this section, we describe an overview of our tool construction approach. The goal of our approach is to identify the most suitable set of objects that can be combined to construct a tool for performing a specified input action (called the *reference action*). In order to do so, we introduce a multi-objective function that effectively scores the fitness of different object combinations for performing the specified action. There are two methods that have been previously proposed for optimizing multi-objective functions, namely, Pareto method and Scalarization [29]. In this thesis, we perform multi-objective optimization through scalarization. Scalarization combines the different objectives in the multi-objective function through a weighted summation. This allows us to control the weights associated with each objective function to determine their relative dominance, unlike the pareto method. As described by Gunantara, *"A large weight that is given to an objective function shows that said function has a higher priority compared to the ones with a smaller weight."* [29].

A general multi-objective function Φ, that is optimized through scalarization can be expressed as a weighted sum over a set of k objectives or features, denoted by $\phi_1, ..., \phi_k$ [29]. In the context of tool construction, for a given tuple of candidate objects $T_i \in T$:

$$\Phi(T_i) = \lambda_1 * \phi_1(T_i) + \lambda_2 * \phi_2(T_i) + ... + \lambda_k * \phi_k(T_i)$$

We show that reasoning about three features of the candidate objects in T_i, namely, ***shape***

(ϕ_{shape}), ***materials*** (ϕ_{mat}), and ***attachments*** (ϕ_{att}), enables the robot to effectively explore the space of possible object configurations. We define *attachments* as the locations at which objects can be attached together. Our work introduces a supervised learning based framework for computing each of the three objective functions, that is computationally scalable as number of candidate objects increases. Note that, we also use the terms "objective" and "scores" interchangeably in this dissertation.

4.2.2 Computational Approach

Our computational approach for tool construction involves four key steps:

- *Workspace Segmentation*: Our pipeline begins with point cloud segmentation, which enables the system to identify the candidate objects in the robot's workspace. We use plane subtraction and Sample Consensus Segmentation (SAC)[3] to identify the candidate objects available to the robot using RGB-D data from an overhead camera. We denote the resulting candidate objects as $O = \{o_1, o_2, ..., o_n\}$.

- *Shape Scoring*: The shape scoring algorithm, evaluates the visual appropriateness of the candidate objects and assigns a corresponding shape score. Depending on the type of descriptor used (parametric or non-parametric), we present two classes of shape scoring approaches in the following sections. Given a list of tuples consisting of all possible candidate configurations generated through the permutation ${}^nP_m$ of candidate objects in O, both shape scoring methods output a score indicating the shape fitness of the objects in each tuple.

- *Material Scoring*: The material scoring algorithm, evaluates the material appropriateness of the candidate objects and assigns a corresponding material score. Given a list of tuples consisting of all possible candidate configurations generated through the

[3]The implementation was provided by the PCL library

permutation ${}^{n}P_{m}$ of candidate objects in O, the material scoring approach outputs a score that indicates the material fitness of the objects in each tuple.

- *Attachment Scoring*: The shape and material scores discussed above do not indicate whether the objects can be successfully attached to construct the tool. Given the tuples from the previous step, our attachment scoring algorithm evaluates whether the candidate objects can be attached via three different modes of attachment. The algorithm outputs a score, which combined with the shape and material scores through weighted summation, is used to generate a final ranking of the object configurations indicating the best configuration for the tool construction.

- *Tool Validation*: Given the final ranking of object configurations, the robot constructs the tools by joining the objects specified by the best-rated configuration, using the output attachment locations and attachment type. The robot then evaluates the constructed tool for its task suitability by applying the desired action on the tool. In this work, we assume that the robot can observe whether the tool succeeded, and that the action trajectory is pre-specified. Alternatively, the action trajectory could be learned from demonstration [75] including, if necessary, adapting the original action to fit the dimensions of the new tool [76, 77]. If tool construction fails or the tool fails at performing the specified action, the robot continues to iterate through the ranked list of tuples until a successful tool is found or the list is exhausted.

4.2.3 Chapter Organization

For the remainder of this chapter, we break down the multi-objective function into its component functions discussed in each section, in the order below:

- **Parametric shape reasoning with predefined attachment points:** (section 4.3) This section focuses on computing the parametric shape scoring objective, and attachment objective when the attachment points are pre-specified.

- **Non-parametric shape and attachment reasoning:** (section 4.5) This section focuses on computing the non-parametric shape scoring objective, and the attachment scoring objective when attachment points are not pre-specified.

- **Shape, material and attachment reasoning:** (section 4.7) This section focuses on computing the material scoring objective, and combining it with the shape and attachment objectives. This section presents the final multi-objective function that is used for performing tool construction.

4.3 Shape Scoring for Tool Construction With Predefined Attachment Points

In this section, we discuss the computation of a parametric shape score that is used to identify suitable objects for tool construction from their parametric shape representations. For attachments, prior work has defined the different joint types commonly observed between the parts of a tool, namely, fixed joints, revolute joints, and prismatic joints [78]. In this section, we demonstrate fixed joints achieved by attaching objects using magnets that are present on them, given a predefined library that specifies the locations of the magnets for each candidate object, as in [79].

4.3.1 Parametric Shape Scoring

In order to compute the parametric shape score, our approach takes a reference tool point cloud as input, and seeks to match candidate objects to the components of the reference tool point cloud. The reference tool is first segmented into its key components, e.g., head and handle for a hammer. The resultant reference tool components are denoted as an ordered tuple $R = (r_1, r_2, ..., r_m)$. Our reference tool model is obtained from the ToolWeb dataset [11], and we use Triangulated Surface Mesh Segmentation[4] to segment the reference tool.

Given the reference tool components and candidate objects available for construction, we compute parametric shape descriptors for the components and objects, by fitting Su-

[4]The implementation was provided by the CGAL library

Algorithm 1: Parametric Shape Scoring and Predefined Attachments

input : importance weights $\lambda_1, \lambda_2, \lambda_3, \lambda_4$
SQ params, $T = permute(O, m)$

output: T^*, Att

$E = [\,]$, $Att = [\,]$
for $i \leftarrow 1$ **to** $|T|$ **do**
  for $j \leftarrow 1$ **to** m **do**
    $e_{shape}(T_i) \stackrel{+}{=} |shape(r_j) - shape(T_{ij})|$
    $e^{T_i}_{scale} \stackrel{+}{=} |scale(r_j) - scale(T_{ij})|$
    for $k \leftarrow 1$ **to** m **do**
      if $k \neq j$ **then**
        $e^{T_i}_{ratio} \stackrel{+}{=} |rel(r_j, r_k) - rel(T_{ij}, T_{ik})|$
    end
  end
  $\phi^{para}_{shape}(T_i) = \lambda_1 e^{T_i}_{scale} + \lambda_2 e^{T_i}_{shape} + \lambda_3 e^{T_i}_{ratio}$
  $\phi_{att}(T_i), A^{T_i}_{closest} = AttachmentFit(T_i)$ *// Described in Algorithm 2*
  $\Phi_{cons}(T_i) = \phi^{para}_{shape}(T_i) + \lambda_4 \phi_{att}(T_i)$
  $E.append(\Phi_{cons}(T_i))$
  $Att.append(A^{T_i}_{closest})$
end
$T^* = sort(T, E)$ *// Sort T based on E in descending order*
return T^*, Att

perquadric (SQ) models. Our full algorithm is shown in algorithm 1. Each SQ model has 13 parameters: 3 for scale in each dimension, 2 for shape variance, 3 for Euler angles, 2 for tapering parameters, and 3 for the central point or mean. For the SQ fitting, we perform non-linear optimization using Levenberg-Marquardt algorithm to find the best fit SQ parameters [11]. Then the parametric shape score is computed by considering the following:

- per-component ***shape*** fit, based on absolute difference of the two SQ parameters indicating the shape variance (denoted by a 2D vector $shape$) between the candidate objects and the corresponding reference components.

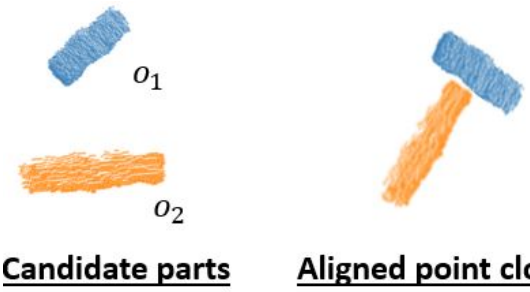


Figure 4.1: Figure showing the alignment of two object point clouds for constructing a hammer. The relative orientations are computed using PCA.

- per-component ***size*** fit, based on absolute difference of the 3 scaling parameters (denoted by a 3D vector $scale$) between the candidate objects and their corresponding reference tool components.

- pairwise component ***proportionality*** fit, calculated based on the absolute difference of the relative scale ratios (denoted by rel) between the reference tool components and the candidate objects, e.g., hammer handle may be $3\times$ longer than the head.

Let, $e^{T_i}_{shape}$, $e^{T_i}_{scale}$ and $e^{T_i}_{ratio}$ denote the shape, size and proportionality fit of the candidate objects in T_i, then the final parametric shape score $\Phi^{para}_{shape}(T_i)$ is computed as (also shown in algorithm 1, Line 11):

$$\phi^{para}_{shape}(T_i) = \lambda_1 e^{T_i}_{scale} + \lambda_2 e^{T_i}_{shape} + \lambda_3 e^{T_i}_{ratio} \tag{4.1}$$

The list of importance weights λ_{1-3}, and the SQ parameters of the candidate objects and reference tool components obtained from the non-linear optimization are provided as inputs into algorithm 1.

4.3.2 Attachment Scoring With Predefined Attachment Points

The attachment scoring indicates whether the objects in T_i can be attached appropriately to construct the desired reference tool. The attachment algorithm is shown in algorithm 2. The process begins by aligning the candidate objects in T_i in a configuration consistent with

the reference tool (line 2). We use Principal Component Analysis (PCA) to orient object point clouds w.r.t. the reference tool, resulting in a set of alignments T_i' (See Figure 4.1 for example of point clouds aligned for a constructing a hammer). We approximate the intersections of the point clouds in each alignment by calculating the centroid of the closest points between the point clouds (line 3). The resultant set of centroids, P, is the candidate list of attachments we want to make, referred to as target attachment locations. The target attachment locations indicate points of *desired* attachment between the candidate objects.

Once the target attachment locations are computed, the attachment score encodes the ***proximity of available attachment points*** to the target attachment locations based on Euclidean distance. Thus, the attachment score is computed based on the Euclidean distance between points in P and the closest attachment locations on each object o_j, in each alignment $t_i \in T_i'$ (lines 5-9). If an object in t_i is known to have no attachment points, then $\phi_{att}(T_i) = -\infty$.

4.3.3 Final Score Computation For Tool Construction

Each of the above metrics produces a value indicating the magnitude of deviation of the candidate object combinations from the reference tool, in terms of shape and attachment capabilities. We then compute the final multi-objective function as a weighted sum of the shape and attachment scores using λ importance weights[5]. The final score that is used for tool construction is as follows:

$$\Phi^{cons}(T_i) = \lambda_1 * \phi^{para}_{shape}(T_i) + \lambda_4 * \phi_{att}(T_i) \tag{4.2}$$

Owing to the size of the configuration space, we use brute force optimization to compute the scores for all object combinations. The final list of candidate object configurations are then sorted by their associated score, from lowest to highest. The output of algorithm 1 is a list of candidate builds T^* sorted by Φ^{cons}, and a list of attachment points Att to use for

[5] λ terms were manually chosen in this work

Algorithm 2: Attachment Scoring

input : candidate objects T_i, attachments A

output: $\phi_{att}(T_i)$, $A_{closest}$

$\phi_{att}(T_i) = -\infty$, $A_{closest} = [\,]$
$T_i' = Align(T_i, R)$ // *pose and orient objects in T_i in ref. to R*
$P = ComputeIntersections(T_i')$ // *Average intersection of objects in T_i*
if $A \neq \varnothing$ **then**
  foreach $t_i \in T_i', o_j \in t_i$ **do**
    $a = ClosestAttachments(P, o_j, A)$
    $\phi_{att}(T_i) += \|P, a\|$ // *Euclidean distance to desired attachments*
    $A_{closest}.append(a)$
  end
else
  return $\phi_{att}(T_i), P$ // *Attachments unknown*
end
return $\phi_{att}(T_i), A_{closest}$

combining the objects when physically constructing the tool.

4.3.4 Exploratory tool construction with Unknown Attachments

If no object attachment information is known, the set P is used to guide the robot in constructing the tool. Specifically, the robot attempts to attach objects at the locations specified by P, and then verifies whether attachment was successful before proceeding (see accompanying video[6]). This search process enables the robot to explore possible attachments until a successful combination is found, however, leading to more tool construction attempts.

4.4 Evaluation - Parametric Shape Scoring with Predefined Attachments

To validate our tool construction approach, we constructed three tools: a hammer, spoon and spatula. Each tool consisted of two components ($m = 2$), and the robot was given

[6]https://www.youtube.com/channel/UCxnm8iu1TS75YNXcAiI-nEw

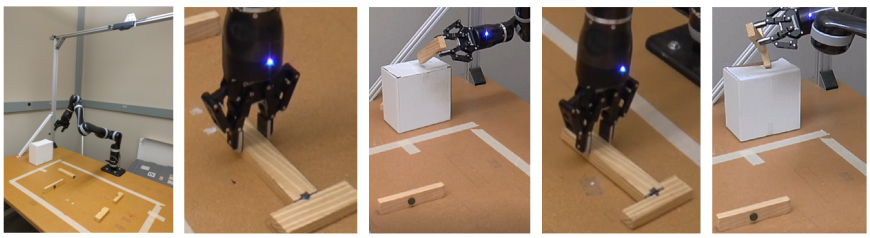
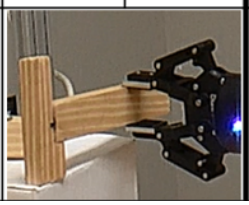
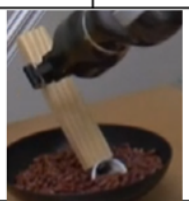


Figure 4.2: Hammer construction, first using objects C and B (failure due to tool breaking apart on impact), and second using parts D and B (success).

$n = 4$ candidate objects. Reference tool models were acquired from the ToolWeb dataset [11]. The manually-set weight parameters $\Lambda = \{1, 1, 5, 5\}$ (shape, scale, ratio, attachment) worked well across all three tasks. We use magnets as attachments for the objects, and seek to validate the performance of our approach both when attachment points are known and unknown. All three tasks, and their results, are summarized in Table 4.1, including: the reference tool used, list of available candidate objects, the number of possible configurations of objects given the attachment points, the object combinations that were attempted and that succeeded (shaded, bold) for each build, along with their corresponding final error value, the number of attempts the robot required to build the tool, and the final constructed tool. Each tool design has a single working configuration, but tests different aspects of our approach. We first describe our results for the case of known attachments, and then unknown attachments. The average computation time across both cases was 15.44 seconds. An example of the tool construction cycle is also shown in Figure 4.2.

With known attachments, we note several key aspects of our approach. First, the robot is able to **validate constructions beyond only geometric fitness of the objects**. Thus, the physical validation of the tool construction proves useful in eliminating constructions that are visually appropriate, but do not function in the real-world. Second, The spoon construction demonstrates the robot's capability to reason about i) **symmetric constructions** (Table 4.1, objects D+B ranked first and second due to symmetric attachments) and ii) **absence of attachments** (scoop with no magnets is ranked poorly). Third, the spatula construction demonstrates the robot's capability to i) **create tools that are diverse** in terms

Table 4.1: For the 3 reference tools, we show the candidate objects and total possible configurations. "Ranking" shows object combinations ranked from best to worst. The bold and shaded scores correspond to the final working tool. Number of physical attempts are also shown for each case.

REF. TOOL	HAMMER		SPOON		SPATULA	
PARTS	 A B C D		 A B C D		 A B C D	
# TOTAL CONFIGS	12		18		26	
ATTACH POINTS	KNOWN	UNKNOWN	KNOWN	UNKNOWN	KNOWN	UNKNOWN
RANKING e_{const} ↓	C+B (1.29)	C+B (0.75)	**D+B (0.56)**	C+A (0.28)	D+B (0.51)	D+B (0.28)
	D+B (1.54)	C+A (0.81)	D+B (0.56)	C+B (0.33)	**D+C (0.75)**	D+A (0.41)
	A+B (1.84)	**D+B (0.82)**	D+A (0.79)	D+A (0.53)	D+C (0.75)	**D+C (0.47)**
	C+D (1.92)	D+A (0.83)	D+A (0.79)	**D+B (0.61)**	D+B (0.99)	A+B (0.97)
	C+A (1.94)	C+D (0.9)	A+B (1.6)	C+D (0.86)	D+C (1.11)	A+C (1.01)
# OF ATTEMPTS	2	6	1	14	2	17
FINAL CONSTR.						

of geometry but still accomplish the desired function (constructed spatula differs from the reference tool), and ii) reason about **multiple attachment configurations** (object combinations D+C repeat for the different attachments).

With unknown attachments, the results in Table 4.1 highlight key differences in the construction process compared to known attachment locations. When the robot does not have knowledge of attachment locations a priori, the construction process is guided by the target attachment locations. As a result, the robot attempts multiple attachments per configuration. For example, the solution for the spoon was found using the fourth object combination (D+B) after 14 construction attempts. Thus, identifying a valid combination

of objects using shape information alone is significantly more challenging than when attachment locations are also known. However, in all cases the robot successfully identifies a solution within the first 4 configurations (out of possible 12, 18 or 26), validating that the **final ranking is successful at guiding the search**, exploring only a small subset of all possible object configurations (on average, only 9.9% of all possible object configurations are physically attempted).

Our results in this section provide preliminary insights regarding the benefits of using combined shape and attachment scoring for performing tool construction, enabling the robot to efficiently construct tools while exploring only a small percentage of all possible object configurations. However, in the following sections, we discuss how the computational scalability of the shape scoring method can be improved through non-parametric shape scoring. Further, we introduce two new types of attachment, and present techniques that enable the robot to autonomously reason about potential attachments.

4.5 Shape and Attachment Reasoning

In this section, we discuss the computation of a non-parametric shape score that is used to identify suitable objects for tool construction from non-parametric shape descriptors. Further, we introduce an approach for computing the attachment score when the attachment locations are not specified a-priori. We also introduce two previously unexplored techniques for attaching objects for tool construction, namely, *pierce attachment* (piercing one object with another, e.g., foam pierced with a screwdriver), and *grasp attachment* (grasping one object with another, e.g., a coin grasped with pliers). For completeness, we also include *magnetic attachment* from our previous framework (section 4.3), for a total of three attachment modalities. In our evaluation, we seek to compare parametric and non-parametric shape scoring performances, and highlight the scalability and parameter tuning challenges associated with the parametric approach.

Algorithm 3: Non-parametric Shape Scoring

input : action; $T = permute(C, m)$

output: T^*, Att

$E = [\,], Att = [\,], Type = [\,]$
for $i \leftarrow 1$ **to** $|T|$ **do**
    $\phi_{shape}(T_i) = ShapeFit(T_i, action)$
    $t_{att} = AttachType(T_i)$ *// Attachment type: grasp, pierce or magnetic*
    $\phi_{att}(T_i), A^{T_i}_{closest} = AttachmentFit(T_i, t_{att})$ *// As described in algorithm 4*
    $\Phi_{cons}(T_i) = \phi_{shape}(T_i) + \phi_{att}(T_i)$
    $E.append(\Phi_{cons}(T_i))$
    $Att.append(A^{T_i}_{closest})$
    $Type.append(t_{att})$
end
$T^* = sort(T, E)$ *// Sort T based on E in descending order*
$Att = sort(Att, E)$
$Type = sort(Type, E)$
return $T^*, Att, Type$

4.5.1 Non-parametric Shape Scoring

In order to compute the non-parametric shape score, our approach takes a reference action as input (e.g., "hit" or "flip"), as opposed to a reference tool point cloud as with the parametric shape scoring approach (described in section 4.3). As we demonstrate in our results, this enables us to construct a broader range of tools without conforming to the shape specifications of a particular reference tool. Hence, our algorithm seeks to accomplish a target *action* as opposed to match a reference tool [20]. Note that this approach is consistent with level 1: object improvisation. We present a framework of our approach in Figure 4.3.

Given a reference action as input, the shape scoring module seeks to predict the shape fitness of the candidate objects and output a corresponding fitness score. This is indicated by the $ShapeFit()$ function in algorithm 3. In this work, we consider the constructed tools

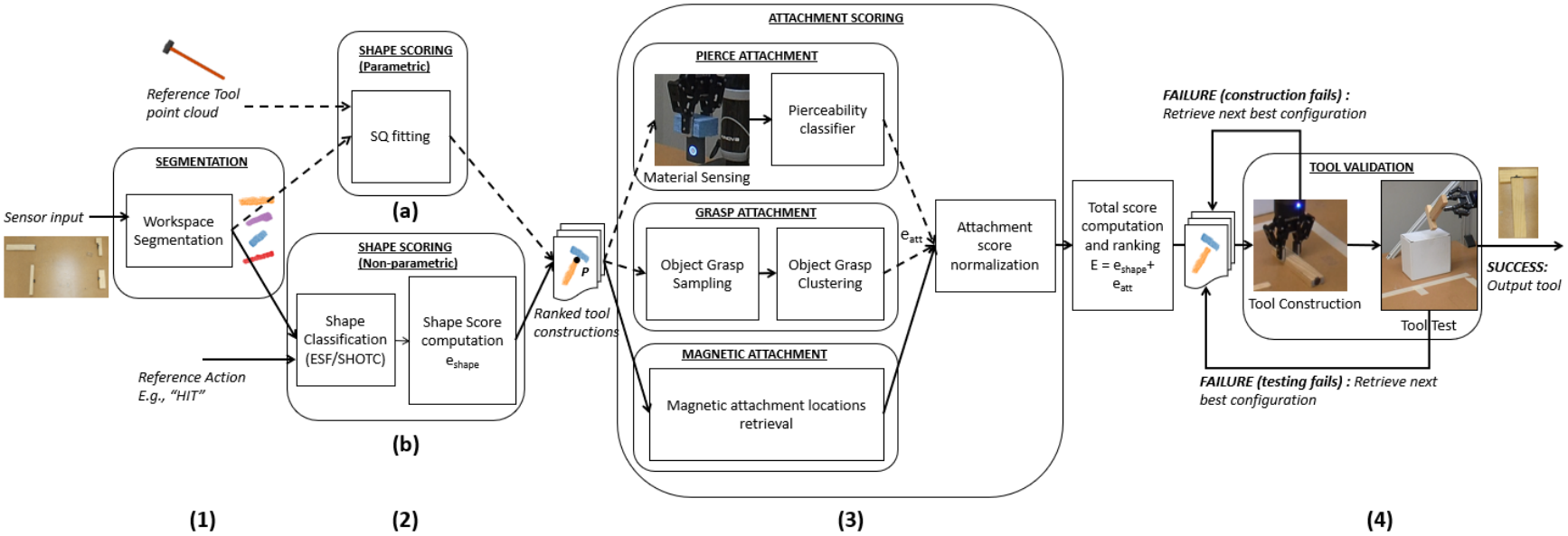


Figure 4.3: Overview of our tool construction approach highlighting the four key steps involved: segmentation, shape scoring, attachment scoring, and tool validation. Solid lines denote the path taken for the example shown, dashed lines represent alternate paths. The framework also denotes parametric shape scoring module from the previous section. We denote the scores with the letter 'e'.

to have action parts and grasp parts[7]. Hence, the set of objects T_i consists of two objects, i.e., $|T_i| = 2$. Further, the ordering of the objects in T_i indicates the correspondence of the objects to the action and grasp parts of the constructed tool, respectively.

For shape scoring, we seek to train models through supervised learning, that can predict whether an input point cloud is suited for performing a specific action. We frame this as a binary classification problem. We represent the candidate object point clouds using two types of non-parametric shape descriptors, namely, Ensemble of Shape Functions (ESF) [52], and Signature of Histograms of Orientations at Centroid (SHOTC) [9]. We seek to compare the performance of the two descriptors for tool construction, in our evaluation. Note that, ESF is a 640-D vector, and SHOTC is a 352-D vector.

Given the shape descriptors, we then train independent neural networks that take an input ESF or SHOTC feature, and output a binary label indicating whether the input shape feature is suited for performing a specific action. Thus, we train separate neural networks, one for each action. A key advantage of this approach is that for new actions, additional networks can be trained independently without affecting the other existing networks. In this work, we utilize five different reference actions: "*Hit*", "*Contain*/*Scoop*", "*Screw*", "*Flip*" and "*Squeegee*". Additionally, we consider a supporting function: "*Handle*", which refers to the tools' grasp part. Thus, the first five correspond to the action parts of the tool which is combined with a "handle" or grasp part for constructing the final tool. The neural network architectures used for ESF and SHOTC both consist of a single hidden layer, with 426 and 235 units respectively. We use ReLU activation, with a sigmoid in the final layer, and apply Stochastic Gradient Descent for training.

For training each action network, we generated a dataset consisting of 3D point clouds from the ToolWeb dataset [10] and other online sources[8]. The models were segmented[9] into their action and grasp parts – e.g., a hammer from the dataset is segmented to its hammer

[7]This covers the vast majority of tools [74, 11]

[8]We used 3dWarehouse and tf3dm

[9]Approach used https://github.com/pauloabelha/batch_segmentation

Table 4.2: Validation accuracies of the prediction models for each tool type

Action	**ESF**	**SHOTC**
Hit	95%	80.5%
Screw	95%	53.3%
Flip	97.5%	69.1%
Squeegee	70%	95%
Contain	92.4%	93%
Handle	90.6%	83.3%

head (action part) and handle (grasp part) – and each part was added to the dataset of its respective type, i.e., 'hit' and 'handle'. This process resulted in six datasets corresponding to the five action types and the 'handle' set, with roughly 32 models per class, for a total of 196 point clouds. For each point cloud, we extract ESF and SHOTC features and train each neural network. The point clouds belonging to the particular class or function are treated as positive examples, and the point clouds from other classes are used as negative examples during training (corresponding 10-fold cross validation accuracies shown in Table 4.2).

Given a reference action a, and a tuple of candidate objects T_i, we can compute the shape score for T_i using the trained networks. Recall that the ordering of the objects within the tuple indicates correspondence to the action or grasp parts of the constructed tool. Let $\mathcal{K}$ denote the set of objects in T_i that are candidates for the action parts of the final tool, and let $T_i - \mathcal{K}$ be the set of candidates for the grasp parts. Then the shape score $\phi_{shape}(T_i)$ is computed by using the trained networks as follows:

$$\phi_{shape}(T_i) = \prod_{o_j \in \mathcal{K}} p(action|o_j) \prod_{o_j \in T_i - \mathcal{K}} p(handle|o_j) \tag{4.3}$$

Where, p is the prediction confidence of the corresponding network. Thus, we combine prediction confidences for all action parts and grasp parts. For example, if the specified action is "hit" and T_i consists of two objects (o_1, o_2), then $\phi_{shape}(T_i) = p(hit|o_1) * p(handle|o_2)$.

Algorithm 4: Attachment scoring for pierce, grasp, and magnetic attachments

input : candidate objects T_i, attachment type t_{att}

output: $\phi_{att}(T_i)$, $A_{closest}$

$\phi_{att}(T_i) = -\infty$, $A_{closest} = [\,]$, $A^{T_i} = \varnothing$

$T_i' = Align(T_i)$

$P = ComputeIntersections(T_i')$ *// Average intersection of objects in T_i*

if $t_{att} = `pierce'$ **then**

  if $isPierceable(T_i)$ **then**

    $A^{T_i} = P$

else if $t_{att} = `grasp'$ **then**

  $A^{T_i} = GraspSample(T_i)$

else if $t_{att} = `magnetic'$ **then**

  $A^{T_i} = userInput(T_i)$ *// Predefined locations for magnetic attachments*

if $A^{T_i} \neq \varnothing$ **then**

  foreach $t_i \in T_i', s_j \in t_i$ **do**

    $a = ClosestAttachments(P, s_j, A^{T_i})$

    $\phi_{att}(T_i) \stackrel{+}{=} \|P, a\|$ *// Euclidean dist to P*

    $A_{closest}.append(a)$

  end

else

  return $\phi_{att}(T_i), P$

end

$\gamma = -max(\phi_{att}(T_i))$ *// normalizer*

return $\phi_{att}(T_i)/\gamma, A_{closest}$

4.5.2 Attachment Scoring

To evaluate whether the parts can be attached appropriately to accomplish the specified reference action, we compute an attachment score. The attachment scoring algorithm is shown in algorithm 4. Given a set of attachment locations, the process of computing the attachment score follows as described in subsection 4.3.2. We begin by aligning the object point clouds in T_i using PCA, in a configuration consistent with prototypical tools used for the specified action. This is used to compute the target attachment locations we wish

to make. The attachment score is then computed as the Euclidean proximity of the target attachment locations, and the attachments facilitated by the candidate objects. We consider three different types of attachments in order of reducing complexity: 'pierce', 'grasp', and 'magnetic' attachments. For pierce and grasp attachments, we assume that objects with pierce capability (screwdrivers and sharp pointed objects), and objects with grasp capability (pliers and tongs) are known a-priori. Note that for pierce and grasp attachments, we seek to autonomously predict the attachment locations. However, for magnetic attachments we use a pre-specified library of attachment locations as described in subsection 4.3.2. Hence, we compute the attachment locations for each attachment type, as described below.

Pierce Attachment

For pierce attachments, we seek to train a model that can predict material pierceability of a given candidate object. We frame this as a binary classification problem. As input, the model takes spectral readings of an object, scanned using a commercially available hand-held spectrometer[10], called the SCiO. The SCiO senses object material properties to return a 331-D real-valued vector indicating the spectral intensities reflected by the object. In prior work, material classification using SCiO was shown to achieve 94.6% accuracy combined with recurrent neural networks [59]. Given a dataset of SCiO measurements from an assortment of objects, we use supervised learning to train a neural network to output a binary label indicating material pierceability. For this work, we assume homogeneity of materials, in that if a material is pierceable, it is pierceable uniformly throughout the object.

For our model, we use a neural network with a single hidden layer of 256 units, and binary output layer. We used the Adam optimizer with ReLU activation layer, and a sigmoid in the final layer. To train our model, we used an existing dataset[11], SMM50, with spectrometer readings for 4 classes of materials: plastic, wood, metal and foam. For each material class, 12 different objects were used, with 50 samples collected per object from

[10]https://www.consumerphysics.com/

[11]Dataset available at https://github.com/Healthcare-Robotics/smm50

scanning different locations of the object. This results in a total of 600 spectrometer readings per class. For each, we provide the pierceability labels. In our case, only 150/600 spectral readings correspond to pierceable objects, i.e., foam objects. Our model yields an accuracy of 98% with leave-one-out cross validation on the SMM50 dataset.

To determine the attachment score during tool construction for the input T_i, the SCiO sensor is used to scan the objects and the corresponding spectral reading is passed to the classifier. If the output label is zero (line 5, algorithm 4, $isPierceable(T_i) = 0$), $A^{T_i} = \varnothing$, since pierce attachment is not possible. If pierceable, $A^{T_i} = P$, assuming homogeneity of material properties allowing the objects to be configured at the desired target attachment locations. The attachment locations A^{T_i} is then used to compute ϕ_{att} based on their Euclidean proximity to P. Note that, since $A^{T_i} = P$, this leads to a very low attachment score (which is preferred), indicating that the objects can be attached exactly as desired, through pierce attachments. However, it is possible to add a fixed cost to pierce attachments in order to make them a less preferable mode of attaching objects, since they can potentially damage the objects that are used for construction.

Grasp Attachment

Grasp attachment is defined as using one object to grasp or hold another object to extend the robot's reach (e.g., grasping a bowl with pliers). We model the grasping tool (pliers or tongs) as an extended robot gripper, allowing the use of existing robot anti-podal grasp sampling approaches [80, 81, 82] for computing locations where the tongs or pliers can grasp other objects. In particular, we use the approach discussed in [80] that outputs a set of grasp locations for a point cloud, given the input parameters reflecting the attributes of the pliers or tongs used for grasping. We cluster the grasp locations (using Euclidean metric) to identify unique grasps, which serve as the grasp attachment locations for the point cloud. As described in [80], without any additional training, the geometry-based grasp sampling approach achieves an accuracy of 73%. To further improve the accuracy, it is possible

to train an object-specific model to identify valid grasps. A key challenge with using a pre-trained model is the need to re-train it for every newly encountered pliers/tongs with differing parameters. This can be inefficient in terms of computational time and resources. Hence, we use the geometry-based grasp sampling without any object-specific refinement.

To determine the attachment score during tool construction for the input T_i, grasps are sampled for the objects (Line 11, $GraspSample(T_i)$) using the existing grasp sampling algorithm[12]. Once sampled, the resultant grasps are returned as potential attachment locations A^{T_i}. The attachment locations A^{T_i} are then used to compute ϕ_{att} based on their Euclidean proximity to P.

Magnetic Attachment

Magnetic attachments are incorporated similar to subsection 4.3.2. We assume the locations of magnets to be provided or predefined, and simply compute the ϕ_{att} score based on the Euclidean proximity of the user input attachment locations (Line 13, $userInput(T_i)$) to P. However, as described in section 4.3, if no part attachment information is known, the target attachment locations are used to guide the robot in constructing the tool.

Final Attachment Score

Let $A^{T_i}(o_j)$ denote the attachment locations present on the object $o_j \in T_i$, computed as described in the above sections. Then, the final attachment score is computed as:

$$\phi_{att}(T_i) = \begin{cases} -\sum_{o_j \in T_i} \left\| P - A^{T_i}(o_j) \right\|, & \text{if attachable} \\ -\infty, & \text{otherwise} \end{cases} \tag{4.4}$$

The negative sign indicates that lower attachment scores are preferred, since we want the attachment locations to be as close as possible to the target attachment locations.

[12]Implementation at https://github.com/atenpas/gpg based on [80]

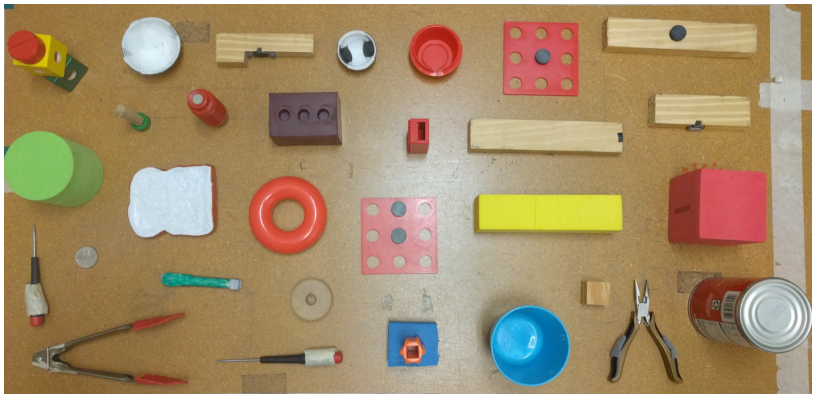


Figure 4.4: The 30 objects used for experimental validation.

4.5.3 Final Score Computation For Tool Construction

Given the shape score from Equation 4.3, and the attachment score from Equation 4.4, we compute the aggregate final score as a weighted sum of $\phi_{shape}(T_i)$, and the normalized $\phi_{att}(T_i)$. We update Equation 4.2, with the non-parametric shape score as follows:

$$\Phi^{cons}(T_i) = \lambda_1 * \phi_{shape}(T_i) + \lambda_2 * \phi_{att}(T_i) \tag{4.5}$$

The list of candidate object configurations are then sorted by their associated combined scores, from highest to lowest score. The resultant ranking is then used by the robot to construct the final tool.

4.6 Evaluation - Non-parametric Shape Scoring with Attachment Reasoning

In this section, we describe our experimental setup and present our results. We validate our approach on the construction of tools for five different actions: 'hit', 'scoop/contain', 'flip', 'screw' and 'squeegee'. Each tool consists of two components ($m = 2$) corresponding to the action part ('hit', 'scoop/contain', 'flip', 'screw', 'squeegee') and grasp part ('handle'). The performance of the algorithm is evaluated for each tool in terms of the final ranking output by our algorithm. The tool models used to compute the target attachment locations, is acquired from the ToolWeb dataset [11]. Our experiments seek to validate three key

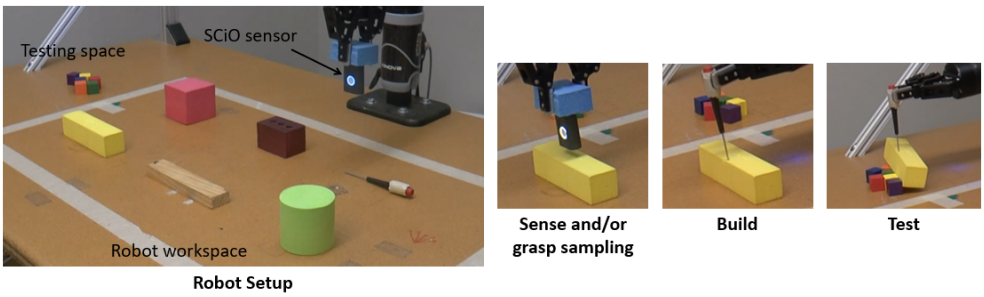


Figure 4.5: Image showing robot setup and steps involved in a typical tool construction cycle. In the case of pierce attachment, the robot uses the SCiO sensor to sense material properties and in case of grasp attachment, the robot samples valid grasps for the object. The robot then builds the tool and tests it by performing the reference action with the tool.

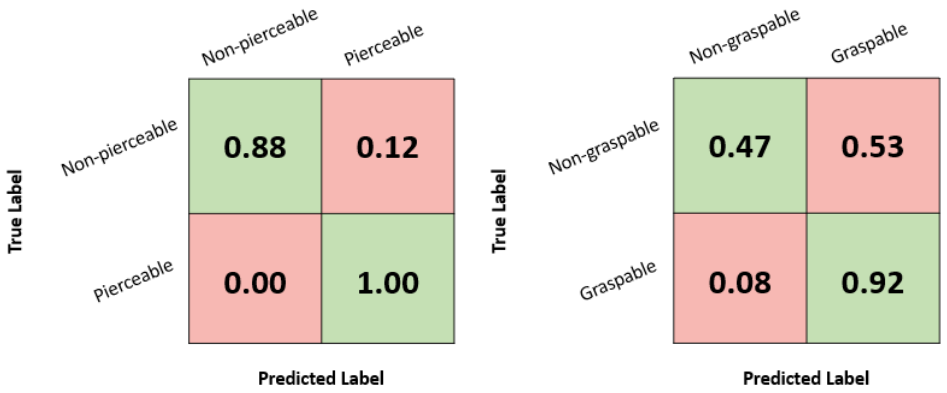
aspects of our approach as follows:

1. *Attachment scoring evaluation*: Performance of the different attachment predictors, namely, pierce and grasp predictors;

2. *Comparison of parametric and non-parametric shape scoring*: Performance of ESF, SHOTC and SQ representation of candidate objects for shape scoring;

3. *Final tool ranking evaluation*: Performance of the overall pipeline in terms of final tool ranking for the five actions described above. We evaluate this for three cases of ablations: Considering only shape scoring, only attachment scoring, and combined shape and attachment scoring.

For all of our following experiments, we use a test set consisting of 30 previously unseen candidate objects for tool construction. These objects consist of metal (8/30), wood (8/30), plastic (9/30) and foam (5/30) objects. However, only 4 of the 5 foam objects are pierceable, while the remaining 26 objects are non-pierceable. Figure 4.5 shows a sample experimental setup and steps involved in the robot tool construction process. During tool construction, the robot begins by scanning the objects using SCiO, and computing the final ranking of object combinations. The robot iterates through the ranking to construct each tool, and validates the construction by performing the reference action with the constructed



(a) (b)

Figure 4.6: Confusion matrixes for (a) pierceability prediction (90% accuracy), and (b) grasp prediction (67% accuracy), over the set of 30 objects.

tool. To overcome manipulation and perception challenges beyond the scope of this work, the available objects were spaced apart and oriented to facilitate tool construction.

4.6.1 Attachment Scoring

In this section, we evaluate the performance of the pierceability and grasp sampling prediction models. For this evaluation, each attachment predictor is tested on the full set of 30 objects, and we report the accuracy indicating the number of objects for which the pierceability or graspability is correctly identified.

The results for the 30 test objects are shown in Figure 4.6a. Non-pierceable objects are classified correctly in 88% of the cases, and a 100% of all the pierceable objects are correctly categorized as pierceable, resulting in an overall accuracy of 90% on the test set for pierceability prediction.

For grasp attachment, the results are shown in Figure 4.6b, with an overall accuracy of 67% on the test set of 30 objects. There are several false positives which potentially affects the ranking of the correct combinations. However, during the tool construction phase, the false positives are eliminated when the robot attempts the actual construction and fails, albeit resulting in a greater number of tool construction attempts on the robot. In the future, grasp prediction could be improved if needed, by using a model that is pre-trained on the

candidate objects [80].

4.6.2 Parametric and non-parametric shape scoring

In this section, we compare the performance of the parametric and non-parametric shape scoring approaches. We use five different sets of four objects (chosen from the 30) for each of the five tools, and report the average results (total 5×5 cases with four candidate objects per case). In each set, we include at least one "correct" combination of objects, and the remainder of the objects are chosen randomly. The correct combination is determined based on external human assessment of the objects. We rank the object combinations using *only* the non-parametric shape scoring (ESF or SHOTC using reference action), or parametric shape scoring (SQ using reference tool), without attachment scoring.

The primary metrics used for our evaluation are i) the final ranking of the correct combinations, and ii) the computation time. We would like the correct combination to be ranked as high as possible, ideally ranked at 1, indicating that it would be the first object combination the robot will attempt to construct. We report the average rank of the correct object combination for each tool (average of 5 builds), the number of builds for which the correct combination was ranked within the top 5 ranks (hits@5), the average number of possible configurations of objects, and the average total computation time. The number of object configurations highlight the state space complexity, and is also used to compute the percentage of total object combinations explored.

The results are shown in Table 4.3. As can be seen by the average rank and hits@5 metrics, SQ and ESF perform somewhat better than SHOTC, although no method dominates all others. However, ESF has significantly better run time performance compared to SQ (average 0.178 seconds for ESF compared to 12.96 seconds for SQ). Hence, the learning based framework is more scalable as the number of candidate objects increase, mitigating the impact of a combinatorial search space. Overall, since ESF outranks SQ and SHOTC on three out of the five tools, and is computationally significantly faster, we use only ESF

Table 4.3: Performance of SQ, ESF and SHOTC (only shape scoring is used) averaged across the five builds for each of the five tools. The shown reference tools are used for SQ computations and reference actions for ESF and SHOTC; Shown in bold are the best performing representations for each action.

REFERENCE TOOL															
REFERENCE ACTION	**HIT**			**SCOOP/CONTAIN**			**FLIP**			**SCREW**			**SQUEEGEE**		
REPRESENTATION	*SQ*	*SHOTC*	*ESF*	*SQ*	*SHOTC*	*ESF*	*SQ*	*SHOTC*	*ESF*	*SQ*	*SHOTC*	*ESF*	*SQ*	*SHOTC*	*ESF*
AVERAGE RANK	3	5	**3**	**3**	5	4	**2**	3	**2**	6	10	**6**	7	**4**	5
HITS@5	4/5	3/5	**5/5**	**5/5**	3/5	4/5	**5/5**	5/5	**5/5**	2/5	0/5	**3/5**	2/5	**4/5**	3/5
AVG. COMPUTATION TIME (seconds)	11.3	1.18	0.18	12.45	1.18	0.2	13.81	1.2	0.18	13.78	1.14	0.17	13.48	1.12	0.16

in our final evaluation in the following section, to compute the final tool construction score.

4.6.3 Final Tool Ranking

In this section, we evaluate our overall tool construction pipeline, combining both shape and attachment reasoning. We use the same sets of objects and evaluation metrics as used in the previous section, again with five builds per reference action. We compare the performance of four object ranking approaches for an ablation study, i.e., using only shape scoring with ESF (S), using only attachment scoring (A), using combined attachment and shape scoring ($S + A$), and using random ranking as a baseline (R) to highlight the problem complexity.

The results of the evaluation are shown in Table 4.4, and the corresponding tool constructions are shown in Table 4.5. The random baseline achieves an average rank of 12, resulting in trials of 56% of candidate object pairs, on average, before finding a working tool solution. Only 24% of the builds are completed in fewer than five tries (hits@5 metric).

In the ESF-only (S) condition, the algorithm achieves an average ranking of 4, and in 80% of trials the correct combination is ranked in the top five (hits@5). On average, only 19% of all the object combinations are explored. These results demonstrate that shape information, even if used alone, is a useful metric for pruning the search space of potential object combinations. In the attachment-only (A) condition, the algorithm also receives an average ranking of 4, and in 92% of trials the correct tool combination is ranked in the top

Table 4.4: Performance of shape reasoning only (*S*), attachment reasoning only (*A*), combined shape and attachment reasoning (*S* + *A*), and random ranking (*R*) conditions for each of the five target actions, averaged over five builds. Shown in bold is the best performing strategy.

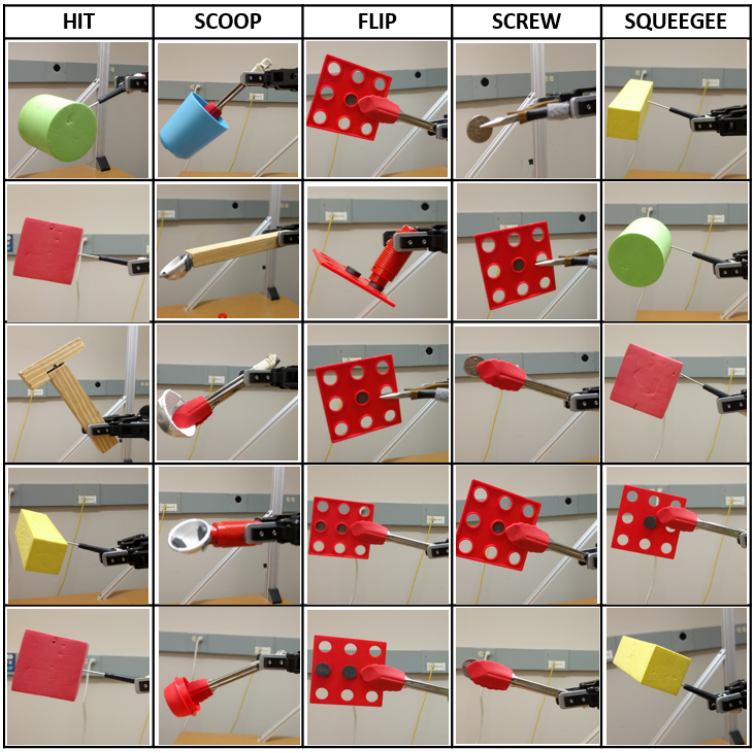
REF. ACTION	HIT				SCOOP/CONTAIN				FLIP				SCREW				SQUEEGEE			
SCORING STRATEGY	*S*	*A*	*S + A*	*R*	*S*	*A*	*S + A*	*R*	*S*	*A*	*S + A*	*R*	*S*	*A*	*S + A*	*R*	*S*	*A*	*S + A*	*R*
AVERAGE RANK	3	3	**2**	11	4	3	**2**	9	2	4	**1**	19	6	6	**4**	10	5	3	**2**	12
HITS@5	5/5	5/5	**5/5**	1/5	4/5	4/5	**5/5**	2/5	5/5	5/5	**5/5**	0/5	3/5	4/5	**5/5**	3/5	3/5	5/5	**5/5**	0/5
PROPORTION OF SPACE EXPLORED	0.17	0.17	**0.12**	0.65	0.2	0.15	**0.1**	0.45	0.07	0.13	**0.03**	0.63	0.24	0.24	**0.16**	0.4	0.27	0.2	**0.11**	0.67
AVG. NUMBER OF POSSIBLE CONFIGS	17				20				30				25				18			

five (hits@5). On average, 18% of the total part combinations are explored. These results show that predicting which objects could be connected together, even without reasoning about the suitability of their shape, again leads to improved performance in the search for the correct object combination, especially in the number of cases the correct objects are ranked in the top five. In the combined shape and attachment ($S + A$) condition, the algorithm achieves an average ranking of 2. In 100% of the cases, the correct object combination is ranked in the top five (hits@5). On average, only 10.4% of the total space of object combinations is explored. These results highlight the complimentary nature of the shape and attachment metrics, and their combined effectiveness in consistently predicting the correct object combination.

4.6.4 Key findings

First, the final approach using combined shape and attachment scoring was **efficient** in that, the robot explored only a small percentage of possible object combinations (average 10.4%), on average requiring only *two* construction attempts per tool. The computational time of the shape modeling component was only 0.178 seconds, indicating that this technique will **scale well** to larger domains. Second, we found that **spectrometer readings were very effective** in predicting the pierceability of objects, yielding an overall accuracy

Table 4.5: Constructed output tools highlighting the diversity of the tool constructions.



of 90%. Third, we found that **shape reasoning with ESF** outperformed SQ and SHOTC in terms of average ranking and computation time. Fourth, either shape or attachment reasoning alone already provided significant guidance in tool creation. However, **combined shape and attachment reasoning** led to the most efficient performance. To the best of our knowledge, this is the first work to demonstrate physical tool construction utilizing multiple attachment types. Table 4.5 shows the diverse tools constructed by our approach.

The results presented here indicate that non-parametric shape scoring through supervised learning with ESF features outperforms parametric shape scoring using SQs, by significantly reducing computational time. This indicates that non-parametric shape scoring scales well to larger number of candidate objects. Further, we introduced two additional, more complex, modes of attaching objects namely, pierce and grasp attachment, in addition to the magnetic attachments from section 4.3, enabling the robot to construct a more diverse set of tools. In our final tool construction pipeline, we use the non-parametric shape scoring and attachment scoring approaches developed in this section.

4.7 Shape, Material and Attachment Reasoning: Final Multi-Objective Function

In the previous sections, we demonstrated that combined non-parametric shape scoring with ESF and attachment reasoning can enable efficient tool construction. In this section, we discuss how we can additionally reason about material properties of available objects in order to improve tool construction performance compared to the previous sections. Our final, complete tool construction pipeline is shown in Figure 4.7.

4.7.1 Material Scoring

Given a reference action and spectral reading of an object as inputs, material scoring seeks to predict the degree to which the spectral reading is similar to that of canonical tools used for performing the action. Prior work has successfully used dual neural networks to predict material similarity between objects from input images [55]. In contrast, we use dual neural networks to compute the material score from input spectral readings of objects.

Dual neural networks consist of two identical networks, each accepting a different input, combined at the end with a distance metric. The parameters of the twin networks are tied, and the distance metric computes difference between the final layers of the twin networks. The networks are trained on pairs of inputs that are of the same or different classes, to discriminate between the class identity of the input pairs. Once the network weights are learned, we use positive examples (i.e., canonical materials) from the training data to learn an *embedding*. The material score ϕ_{mat} is then computed as the similarity of the query spectral reading to the embedding. This enables us to match the input spectral reading to the variety of canonical materials that facilitate an action, rather than conforming to the materials of a specific tool. Here, we assume that the material of the action part of the tool is most critical to performing the action. As a result, we simplify our model by only considering the material of the action part, e.g., we model a knife consisting of a metal blade and plastic handle, as metal.

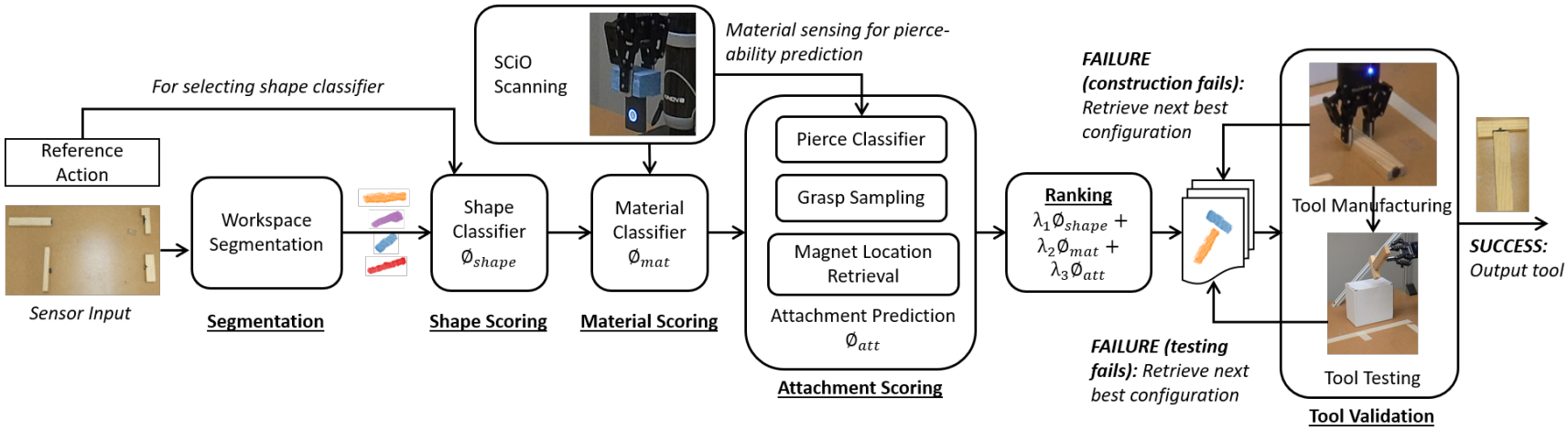


Figure 4.7: The pipeline showing combined reasoning about shape, material and attachment properties of available objects for tool construction

Table 4.6: Table showing the appropriate materials for performing each action, used for generating training pairs for the dual neural network.

Reference Action	**Material (action part)**
Hit	Metal, Wood
Scoop	Plastic, Wood, Metal
Flip	Plastic, Wood, Metal
Screw	Plastic, Metal
Squeegee	Foam
Rake	Plastic, Wood, Metal
Poke	Metal, Plastic

Feature Representation

We use the SCiO handheld sensor, to extract spectral readings for the objects. The SCiO is used to scan the objects to return a 331-D vector of real-valued spectral readings.

Network Architecture

Our model consists of three hidden layers of 426, 284 and 128 units each. We apply tanh activation and a dropout of 0.5 after each layer. The final layer is a sigmoid computation over the element-wise L_1 difference between the third layer of the two networks. We use Adam optimizer with learning rate of 0.001.

Training

To train the dual neural network, we extend the SMM50 dataset used for pierceability prediction with spectral readings corresponding to paper, for a total of five classes of materials: plastic, paper, wood, metal and foam. For our work, we manually identified the most appropriate material classes for different actions, also shown in Table 4.6. We create random pairings of spectral readings, where both materials in the pair are appropriate for the action, or either one is not. Given a set N of training samples, $y(x_i, x_j) = 1$, if both materials are appropriate for a given action (as indicated by Table 4.6), and $y(x_i, x_j) = 0$, if either x_i or

x_j corresponds to an inappropriate material. That is, for "Hit", (metal, metal) and (metal, wood) pairings are both positive examples, whereas (metal, foam) is a negative example. Note that, each pair does not necessarily consist of the same material class. The reason is that, we would like all appropriate material classes for performing a given action, such as metal and wood for "Hit", to be mapped closer in the embedding space, than metal and foam. This allows us to overcome the variance across material classes, learning an embedding space where the desired material classes are closer in distance. Our training minimizes the standard regularized binary cross-entropy loss function as:

$$\mathcal{L}(x_i, x_j) = y(x_i, x_j) \log(\mathbf{p}(x_i, x_j)) + (1 - y(x_i, x_j)) \log(1 - \mathbf{p}(x_i, x_j)) + \lambda |\mathbf{w}|^2$$

The output prediction of the final layer L, is given as:

$$\mathbf{p} = \sigma(\mathbf{w}^T(|h_{1,L-1} - h_{2,L-1}|) + \beta)$$

Where σ denotes the sigmoidal activation function, β denotes the bias term learned during training, and $h_{1,L-1}$, $h_{2,L-1}$ denotes the final hidden layers of the twin networks respectively. The element-wise L_1 norm of the final hidden layers is passed to the sigmoid function. In essence, the sigmoid function computes a similarity between the output features of the final hidden layers of the two twin networks.

Once the network is trained, we learn an embedding using the positive examples (not pairings) from our training set, $x_i^p \in N$, where x_i^p is an appropriate spectral reading for the action. We denote the output of the final hidden layer, for a given input x as, $f(x) = h_{1,L-1}(x)$. We pass each x_i^p through one of the twin networks (since both networks are identical and their weights tied), to map each input into a d-dimensional Euclidean space, denoted by $f(x_i^p) \in \mathbb{R}^d$. We then compute the embedding as an average over $f(x_i^p)$, for all the positive examples x_i^p, where N_p is the total number of positive examples in the training

set for a specific action:

$$\mathcal{D}^{p}_{action} = \frac{1}{N_p} \sum_{i=1}^{N_p} f(x^p_i) \; \forall \; x^p_i \in N$$

We compute the d-dimensional embedding space $\mathcal{D}^{p}_{action}$, using the spectral readings corresponding to appropriate materials as positive examples, $x^p_i \in N$. The computed embedding represents an aggregation of the most appropriate spectral readings in the training set for a specific action.

Prediction

Given the spectral reading corresponding to a candidate object o_j, we compute $f(o_j)$ using our pre-trained model. Let $\mathcal{K}$ denote the set of objects in T_i that are candidates for the action parts of the final tool. Then, ϕ_{mat} is computed as follows:

$$\phi_{mat}(T_i) = \prod_{o_j \in \mathcal{K}} \sigma(\mathbf{w}^T |\mathcal{D}^{p}_{action} - f(o_j)| + \beta) \tag{4.6}$$

This score represents the similarity between material of the candidate object and the embedding, $\mathcal{D}^{p}_{action}$, representative of all the positive examples within the training data.

4.7.2 Final Score Computation

For the tool constructions, the final score is computed through scalarization, as a weighted sum of the non-parametric shape, material and attachment scores. We use the non-parametric score from Equation 4.3 and the attachment score from Equation 4.4. We found uniform weights of $\lambda_1 = 1, \lambda_2 = 1$, and $\lambda_3 = 1$, to work best for tool constructions. Our final score, modifies Equation 4.5 and adds material scoring from Equation 4.6, to compute Φ^{cons} as:

$$\Phi^{cons}(T_i) = \phi_{shape}(T_i) + \phi_{mat}(T_i) + \phi_{att}(T_i) \tag{4.7}$$

As in the previous sections, the final score is used to generate a ranking of object combinations for tool construction. The robot then iterates through each construction in the ranking until a successful construction is found [21].

4.8 Evaluation - Shape, Material and Attachment Reasoning

In this section we evaluate two aspects of our work:

- **Performance of material scoring**: We evaluate our approach to material scoring on a test set of previously unseen spectral readings;
- **Performance of our final tool construction pipeline**: We evaluate the performance of our tool construction approach in comparison to baselines.

Wherever appropriate, we evaluate the statistical significance of our results using repeated measures ANOVA with a post-hoc Tukey's test. The '**' denotes a statistically significant result with $p < 0.01$.

4.8.1 Material Scoring Evaluation

We test three different material scoring approaches on a test set of 58 spectral readings scanned from previously unseen objects belonging to the five different material classes (14 foam, 10 metal, 3 paper, 12 wood and 19 plastic). Each spectral reading is labeled with the action(s) they are appropriate for, corresponding to their material class. We evaluate material scoring approach for six actions: "Hit", "Cut", "Scoop", "Flip", "Poke" and "Rake" We compare our approach (Dual NN) to a feedforward neural network (fNN). We also compare our work to material classification previously proposed by Erickson et al. [59] (Multi-classifier), which is used to predict the class corresponding to a reading, and then matched with our ground truth in Table 4.6.

For our fNN baseline, we evaluated multiple architectures, design choices, and selected the best performing one. Our fNN baseline for material scoring uses four hidden layers

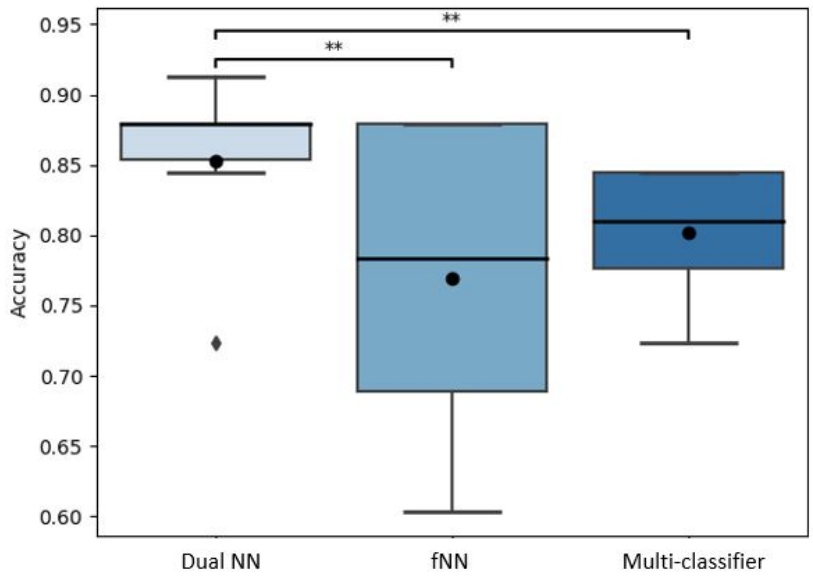


Figure 4.8: Performance accuracy of our material scoring approach for each of the six actions, compared to baselines including Erickson et al. [59] (Multi-classifier).

(64, 64, 32, 32 units), with tanh activation and a dropout of 0.5. We train the model with SMM50 with training samples $x_i \in \mathcal{M}$, where $y(x_i) = 1$ if x_i is an appropriate material for performing the action.

Our results, Figure 4.8, show that our approach outperforms the baselines, with an average accuracy of 85%. While multi-classifier [59] performs almost as well (79.8% accuracy), dual NN is able to capture the *degree* of similarity between candidate and canonical materials, in contrast to only material classification. This allows the dual NN to compute a similarity score, which is beneficial when ranking the object combinations. Shown in Figure 4.9 is a detailed breakdown of our approach on the different material classes. The darker shading denotes the proportion of materials correctly identified for that action by the model. We see that the network largely predicts suitable materials for each action, with some exceptions, such as some foam and paper objects predicted for poke.

4.8.2 Tool Construction Evaluation

In this section, we describe our experimental setup and present the results specifically for our tool construction approach. We validate our approach on the construction of tools for

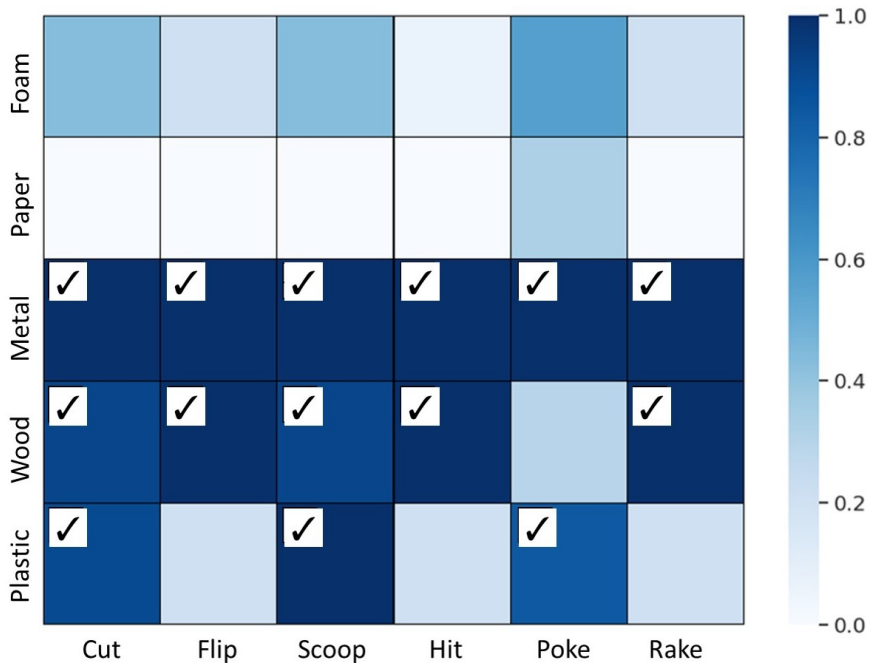


Figure 4.9: Plot showing the proportion of materials predicted by the dual neural network for each action. Checkmarks indicate the materials appropriate for each action (the figure is best viewed in color).

six different actions, encoded as textual inputs: 'hit', 'scoop/contain', 'flip', 'screw', 'rake' and 'squeegee'. Each tool consists of two components ($m = 2$) corresponding to the action part ('hit', 'scoop/contain', 'flip', 'screw', 'rake', 'squeegee') and grasp part ('handle'). The performance of our tool construction approach is evaluated in terms of the final ranking output by our algorithm. We use the final score Φ^{cons} to rank the different constructions. The tool models used to compute the desired attachment location P, is acquired from the ToolWeb dataset [11]. Our experiments seek to validate two aspects of our work:

1. *Final tool ranking evaluation*: Performance of our tool construction approach in terms of the final output ranking, with ablation studies. We use the final score Φ^{cons} to rank the object constructions.

2. *Comparison to prior tool construction approaches*: Performance of our current tool construction approach against the approaches developed in the previous sections[13]

[13]As discussed in the Related Work, we are not aware of any other prior work that demonstrates tool construction using environmental objects.

Figure 4.10: The 58 objects used for experimental validation.

namely, section 4.3 and section 4.5.

For all our experiments, we use a test set consisting of 58 previously unseen candidate objects for tool construction (shown in Figure 4.10). These objects consist of metal (11/58), wood (12/58), plastic (19/58), paper (2/58) and foam (14/58) objects. Only the foam and paper objects are pierceable. During tool construction, the robot begins by scanning the materials of the objects for attachment scoring, followed by ranking and construction of the tools. The robot then tests the constructed tool by using it to perform the desired action, iterating through the ranks until a successful construction is found. To overcome manipulation and perception challenges that are beyond the scope of this work, the available objects were spaced apart and oriented to facilitate grasping.

For the evaluation, we create 10 different sets of 10 objects (chosen from the 58) for each of the six tools, and report the average results (total 10×6 cases with 10 candidate objects per case). We create each set by choosing a random set of objects, ensuring that only one "correct" combination of objects exists per test set. The correct combinations are determined based on external human assessment of the objects.

Final Tool Ranking

We evaluate our overall approach in terms of the final output ranking generated on the test sets of objects described in the previous section. We perform ablation studies to com-

Table 4.7: Table showing results of our ablation studies. Combined shape, material and attachment reasoning (in bold) performs best. Arrows indicate whether lower or higher values are preferred, e.g., lower ranks are preferred.

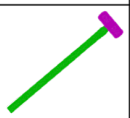
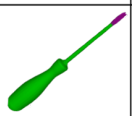
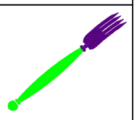
	Shape	Attach	Material	Shape + Att	Att + Mat	Shape + Mat	Shape + Att + Mat	Random
Rank ↓	18.45	13.3	30.37	8.43	11.2	14.19	**5.84**	50.9
Hits@5 ↑	0.45	0.47	0.08	0.42	0.38	0.18	**0.67**	0.03
Rank % ↓	18.08%	13.03%	29.77%	8.26%	10.98%	13.91%	**5.72%**	49.9%

pare performances of shape, material and attachment reasoning for tool construction. For shape scoring, we use non-parametric shape scoring since our experiments in section 4.5 demonstrated that non-parametric shape scoring outperformed parametric shape scoring.

The metrics used in this evaluation consider i) the final ranking of the correct combinations, and ii) the computation time. We would like the correct combination to be ranked as high as possible, ideally ranked at 1, indicating that it would be the first object combination the robot will attempt to construct. We report the average rank of the correct combination for each tool (average of 10 builds), the number of builds for which the correct combination was ranked within the top 5 ranks (hits@5), the average number of possible configurations of objects, and the average total computation time. The number of object configurations highlight the complexity of the state space and is also used to compute the rank% as the fraction of rank over total configuration space.

Table 4.7, shows the overall performance of our approach, and Table 4.8 shows a tool-wise breakdown. From Table 4.7, we see that our final approach combining shape, material and attachment scoring, yields a rank of 5.84, with 67% hits@5, and 5.72% rank%. Hence, we see that there is a significant benefit to combining shape, attachment and material reasoning, in terms of final ranking, rank% and hits@5. Using only shape and attachment also performs well with a rank of 8.43 and rank% of 8.26%, in comparison to the other baselines. All approaches significantly outperform random ranking, which explores roughly half of the entire configuration space (with rank% of 49.9%). In Table 4.8, we show the performance of combined shape, material and attachment reasoning for each action. Also

Table 4.8: Table showing tool-wise breakdown of the combined shape, material and attachment reasoning approach along with the example tools used for computing the target attachment locations.

	Hit	Scoop	Flip	Squeegee	Screw	Rake
Sample tools used to perform each action						
Rank ↓	4.1	3.6	6.7	10.33	5	5.77
Hits@5 ↑	0.8	0.8	0.5	0.55	0.7	0.55
Rank % ↓	4.41%	3.83%	6.93%	10.79%	5.39%	4.76%
Configuration space	93	94	97	96	93	121

shown are some example tools used for the computation of the target attachment locations, P. Overall, our approach using combined shape, material, and attachment reasoning, achieved an average rank of 5.84 across all tool types. Note that the total configuration space for each tool is large (avg. $\approx$ 100 configurations), indicating the complexity of the problem space, and the effectiveness of our combined reasoning approach in ranking the tool construction with a rank% of 5%. Thus, only a small fraction of the total configuration space is explored by our approach. Also note that the approach performed relatively worse on "squeegee" with a rank of 10.33, primarily because none of the available object combinations closely resemble an actual squeegee, making it challenging for tool construction. Our approach achieves an average rank of 5.03 across the remaining tool types.

Comparison to Previous Tool Construction Approaches

We compare our final tool construction approach incorporating material reasoning, to the work described in section 4.3 (denoted by Nair2019a [20]) and section 4.5 (denoted by Nair2019b [21]). We use the same set of objects and evaluation metrics as the previous section, additionally adding the completion rate metric to indicate how many of the total 60 constructions, were successfully found. We mark a tool construction attempt as a failure if either, 1) the correct combination was assigned a score of $-\infty$, e.g., due to incorrect attachment or material predictions or, 2) the approach returned a tool that did not match in

Table 4.9: Table showing performance of our current approach against previous tool construction approaches, [20] and [21].

	Nair 2019a	Nair 2019b	Current
Completion rate ↑	27%	60%	**96.67%**
Rank ↓	8.19	8.45	**5.84**
Hits@5 ↑	0.44	0.37	**0.67**
Rank % ↓	8.03%	8.28%	**5.72%**
Computation time (s) ↓	140.1	14.63	**13.4**

terms of material, e.g., hammers constructed out of foam.

Our results are shown in Table 4.9 and Figure 4.11. As shown in Table 4.9, our current approach outperforms our prior work with a high completion rate of 96.67%, rank of 5.84 and hits@5 of 67%. Hence, there is a significant improvement in the tool construction pipeline with the introduction of material reasoning, reflected by the lower completion rates of the other approaches (27% for Nair2019a and 60% for Nair2019b). Our approach fails at some constructions owing to incorrect pierceability and graspability predictions.

Figure 4.11 shows the diversity of the tool constructions output by our final approach, including several interesting combinations, e.g., combining pliers and coin to create a screwdriver (Construction #10). The symbols at the lower left corner indicate *failed* constructions for each approach. Note that, 91% of the failure cases in our prior approaches were owing to incorrect materials of the constructed tools. Overall, our final tool construction approach is able to effectively reason about materials, resulting in improved quality of constructions over prior work. Additionally, our results demonstrate the capability of combined shape, material and attachment reasoning to construct a diverse set of tools.

4.9 Findings and Contributions

The key findings of this chapter can be summarized as follows:

- Combining shape, material and attachment reasoning leads to significantly improved performance for tool construction when compared to the different ablations.

Figure 4.11: Figure showing a collage of the complete 60 tool constructions in our test set, constructed for six different actions. Note that a small number of experiments (8/60) led to the creation of similar tools due to the availability of objects that could be connected. A symbol on the bottom left of each image indicates that a given approach failed to find the correct construction in that case: ○ : Current work, □: Nair2019b, and △: Nair2019a.

- Non-parametric shape scoring outperforms parametric shape scoring in terms of computational performance, thus scaling to larger number of candidate objects.

While in this chapter we performed brute force optimization which is made possible due to the scalability of our approach, we further note that the proposed multi-objective function (Equation 4.7) can also be used in conjunction with discrete optimization algorithms such as genetic algorithms [83]. However, we found that genetic algorithms were very slow to compute, and hence practically inefficient for performing tool construction. Nevertheless, it is possible to use other discrete optimization algorithms in conjunction with the multi-objective function to perform tool construction.

The contributions of this chapter answer the research question of, *"How can we enable robots to perform tool construction?"*. We find that combined reasoning about shape, materials, and attachment capabilities of objects enables robots to efficiently perform tool construction, and construct a diverse set of tools from various objects.

CHAPTER 5
TOOL SUBSTITUTION

How can we enable the robot to perform tool substitution?

In this chapter, we propose that reasoning about shape and material properties of available objects, can enable robots to effectively perform tool substitution. We first revisit related work and discuss unaddressed challenges in tool substitution. We then present our algorithmic contributions and experimental results. Finally, we summarize our key findings.

5.1 Challenges in Tool Substitution

Some of the key unaddressed challenges in tool substitution that the contributions of this chapter seek to address are as follows:

- **Combined shape and material reasoning for tool substitution:** Prior work in tool substitution has primarily focused on reasoning about shape similarities between objects to identify good substitutes [10, 9]. However, we posit that reasoning about material properties of objects in addition to their shape, is critical for performing tool substitution effectively. For instance, between a metal cup and a foam cup, the metal cup would be a better substitute for a hammer.

- **Limited data for training models for tool substitution:** The limited availability of 3D tool model datasets makes it difficult to train complex model architectures such as PointNet and PointNet++ [84, 85]. For example, the ShapeNet dataset [86] used by PointNet, only has 1/6 tools tested in this work, namely knives. Training well-performing models for predicting tool substitutes, require large datasets which is currently limited for tools.

In contrast to prior work in the field, this chapter contributes a novel approach to reason about *both shape and material* properties of objects, to perform efficient tool substitution. We also present a dataset of tool models gathered from several online sources, to compile a set of $\approx$ 800 tools (also made publicly available[1]).

5.2 Shape and Material Reasoning for Tool Substitution

In this section, we present our tool substitution approach (Figure 5.1 shows an overview). Intuitively, our approach reasons about the degree to which a given point cloud shape and material are similar to that of canonical tools used for performing a given action. This allows us to differentiate and rank the candidate objects based on the similarity score. Each of the n candidate objects in $O = \{o_1, o_2, ...o_n\}$, has an associated ESF feature that is computed, and a spectral scan obtained from the spectrometer. For shape reasoning, we seek to score the shapes of the candidate objects on the degree to which they match the shapes of canonical/normative tools often used for performing the action (the canonical tool models are obtained from existing sources, such as ToolWeb [10]). We use dual neural networks to reason about the similarity between candidate objects and the canonical tools. For material scoring we follow the approach described for tool construction in Chapter 4. For performing shape scoring for tool construction, we used a part-based shape scoring approach where the candidate objects were evaluated separately for the action and grasp parts of the tool. In contrast, we introduce a novel approach for performing shape scoring for tool substitution, called "Joint Shape Scoring", that evaluates the fitness of the candidate objects relative to the *full* tool point cloud, as opposed to evaluating them separately for the action or grasp part of the tool. As we describe in subsection 5.2.4, the joint shape scoring approach can also be extended for performing shape scoring for tool construction.

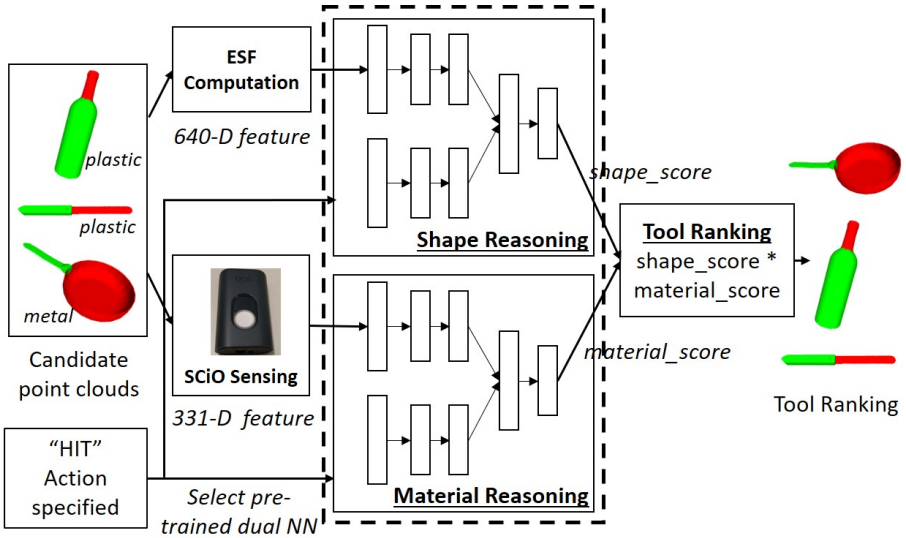


Figure 5.1: The desired action and the ESF and spectral readings for each candidate, are passed through dual networks for shape and material reasoning. The tools are ranked using the combined score (product of shape and material scores).

5.2.1 Joint Shape Scoring

Joint shape scoring takes in a candidate object point cloud and target action, and outputs a score indicating the degree to which the given point cloud is appropriate for performing the reference action. Given a set of actions A, a given candidate object can be appropriate for multiple actions, e.g., a fork could be used for both poking and cutting. Hence, instead of using a single dual network trained for all the actions, we train *separate* networks for each action in A. This also allows new actions to be trained and incorporated into the framework without affecting existing models. Given that $|A| \approx 10$ for most household robots [13, 87], our approach can easily scale to such domains.

We represent the candidate object point clouds using ESF features [52], which is passed as input to our shape network. Our dual neural network architecture consists of three hidden layers of 100, 100 and 25 units each. We apply tanh activation and a dropout of 0.5, after each layer. Our choice of loss function is inspired by prior work in dual neural networks proposed by Koch et al. [88]. Hence, the final layer is a sigmoid computation over the

[1] https://github.com/Lnair1993/Tool_Macgyvering

element-wise L_2 difference between the third layer of each of the two networks, which yields the final output. We use Adam optimizer with learning rate of 0.0001.

Training

To train[2] the dual neural network, we compiled a dataset of 3D tool models from existing online sources, namely ToolWeb [10] and 3DWarehouse (the training dataset used has been made publicly available[3]). For each action, we create random pairings of tools that, based on their shape, can both be used to perform the same action, as well as pairs that cannot (see Figure 5.2). Let N be the set of training samples, then we assume that a pair (x_i, x_j) is positive i.e., $y(x_i, x_j) = 1$, if both x_i and x_j can perform the same action and negative i.e., $y(x_i, x_j) = 0$, when either x_i or x_j is not suited for the action. We minimize the standard regularized binary cross-entropy loss function as:

$$\mathcal{L}(x_i, x_j) = y(x_i, x_j) \log(\mathbf{p}(x_i, x_j)) +$$
$$(1 - y(x_i, x_j)) \log(1 - \mathbf{p}(x_i, x_j)) + \lambda |\mathbf{w}|^2$$

The output prediction of the final layer L, is given as:

$$\mathbf{p}(x_i, x_j) = \sigma(\mathbf{w}^T(|h_{1,L-1}(x_i) - h_{2,L-1}(x_j)|^2) + \beta)$$

Where σ denotes the sigmoidal activation function, β denotes the bias term learned during training, and $h_{1,L-1}$, $h_{2,L-1}$ denotes the final hidden layers of the twin networks respectively. The element-wise L_2 norm of the final hidden layers is passed to the sigmoid function. In essence, the sigmoid function computes a similarity between the output features of the final hidden layers of the two twin networks.

Once the network is trained, we learn an embedding using the positive examples (not

[2]Our models are trained in Keras using Tensorflow Backend.
[3]https://github.com/Lnair1993/Tool_Macgyvering

pairings) from our training set, $x_i^p \in N$, where x_i^p is a canonical tool for the action. We denote the output of the final hidden layer, for a given input x as, $f(x) = h_{1,L-1}(x)$. We pass each x_i^p through one of the twin networks (since both networks are identical and their weights tied), to map each input into a d-dimensional Euclidean space, denoted by $f(x_i^p) \in \mathbb{R}^d$. We then compute the embedding as an average over $f(x_i^p)$, for all the positive examples x_i^p, where N_p is the number of positive examples:

$$\mathcal{E}^p_{action} = \frac{1}{N_p} \sum_{i=1}^{N_p} f(x_i^p) \; \forall \; x_i^p \in N$$

The d-dimensional embedding, $\mathcal{E}^p_{action}$, is computed for each action and serves as our anchor input, matched against the query input to compute a similarity score.

Prediction

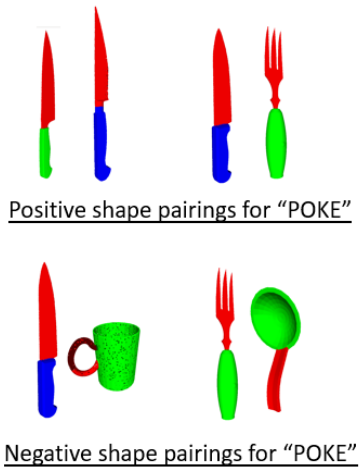
Given the ESF feature of a candidate object, $o_i \in O$, we first compute $f(o_i)$, using our pre-trained model. Then the joint shape score, δ_{shape}, is computed as follows:

$$\delta_{shape}(o_i) = \sigma(\mathbf{w}^T |\mathcal{E}^p_{action} - f(o_i)|^2 + \beta) \tag{5.1}$$

Note that unlike tool construction, only a single object is used for tool substitution. The computed score represents the similarity between the ESF feature of the candidate object and the embedding, $\mathcal{E}^p_{action}$, representative of all the positive examples in the training data.

5.2.2 Material Scoring

For material scoring, we follow the process described in Chapter 4, section 4.7, to train the models and compute the embedding space. Examples of positive and negative material pairings are shown in Figure 5.2. Once the embedding space is computed, we can predict the material score for a given object o_i. Given the spectral reading corresponding to a candidate object $o_i \in O$, we compute $f(o_i)$ using our pre-trained model. Then, ϕ_{mat} is



Action	Appropriate Materials
Hit	Wood, Metal
Flip	Wood, Metal
Cut	Wood, Metal, Plastic
Rake	Wood, Metal
Poke	Metal, Plastic
Scoop	Wood, Metal, Plastic

Figure 5.2: Examples of positive and negative pairings for training the dual networks.

computed as follows:

$$\phi_{mat}(o_i) = \sigma(\mathbf{w}^T|\mathcal{D}^p_{action} - f(o_i)| + \beta) \tag{5.2}$$

This score represents the similarity between material of the candidate object and the embedding, $\mathcal{D}^p_{action}$, representative of all the positive examples within the training data.

5.2.3 Final Score Computation

The final score for tool substitution is computed as a weighted sum[4] of the shape and material scores. We empirically determined uniform weights of $\lambda_1 = 1$ and $\lambda_2 = 1$ to work best. Our final score for tool substitution, Φ^{subs}, is computed as follows.

$$\Phi^{subs}(o_i) = \delta_{shape}(o_i) + \phi_{mat}(o_i) \tag{5.3}$$

Once the final score is computed, the candidate objects are ranked from highest to lowest final scores. The highest ranked object corresponds to the best substitute.

[4]We found no statistically significant difference in tool substitution performance between computing the final score as a weighted sum vs. product of the shape and material scores

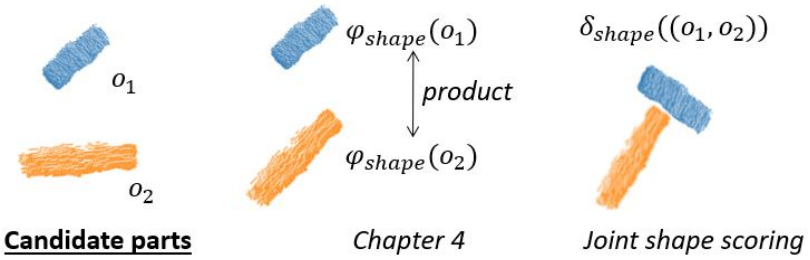


Figure 5.3: Figure highlighting the two types of shape scoring. From Chapter 4, the candidate parts are scored independently, and combined into a single score as a product of their independent scores. For joint shape scoring, the composite object is scored.

5.2.4 Extension to Tool Construction

The shape scoring methodology described in this chapter, can also be applied for scoring tool constructions. For tool construction, given an input tuple of objects $T_i = \{o_1, o_2, ...o_m\}$, we begin by aligning the components in T_i in a configuration consistent with prototypical tools used for the specified action. In order to retrieve this configuration, we sample one random tool from the dataset that was used for training the shape scoring model, corresponding to the specified action. Further, we use Principal Component Analysis (PCA) to orient the object point clouds in T_i with respect to the example tool. Recall that this process is also performed for aligning the point clouds for attachment scoring in Chapter 4. The aligned point cloud is then passed as input to the dual network to compute the shape score:

$$\delta_{shape}(T_i) = \prod_{o_i \in T_i} \sigma(\mathbf{w}^T |\mathcal{E}^p_{action} - f(o_i)|^2 + \beta) \tag{5.4}$$

Figure 5.3 shows an example of the aligned point cloud. The joint shape scoring method effectively treats constructions as substitutes. We will revisit joint shape scoring for tool construction in the context of arbitration in Chapter 6.

5.3 Evaluation

In this section, we evaluate our tool substitution approach, for six actions: "Hit", "Cut", "Scoop", "Flip", "Poke" and "Rake", with five material classes: "Metal", "Wood", "Plastic", "Paper" and "Foam". Our experiment seeks to validate:

- **Performance of shape scoring**: We evaluate our approach for shape scoring on a testing dataset of previously unseen object models;
- **Performance of combined shape and material reasoning for tool substitution**: We evaluate our final approach for tool substitution, comparing it with using only shape scoring, only material scoring, and random ranking baselines. We test our model on a set of partial point clouds and spectral readings of real-world objects.

We evaluate the statistical significance of our results using repeated-measures ANOVA and post-hoc Tukey's test. The '**' denotes a statistically significant result with $p < 0.01$.

5.3.1 Performance of Shape Scoring

We test three different shape scoring approaches on a previously unseen test set of 30 object models, collected from 3DWarehouse. The test set consists of uniquely shaped objects and allows us to measure the generalization capability of the models. We label each object in the set with the action(s) it is appropriate for (based on shape only), which acts as our ground truth label. We compare our approach (Dual NN) to a feed-forward neural network (fNN), and the approach previously proposed by Abelha et al. using SQs [10], as baselines for shape scoring[5].

For our fNN baseline, we evaluated multiple architectures, design choices, and selected the best performing one. The best model architecture uses three hidden layers (100, 100 and 25 units each), with tanh activation and dropout of 0.5 after each layer, with sigmoid

[5] As described in the intro of this chapter, complex model architectures such as PointNet and PointNet++ [85, 84] require vast amounts of training data that are not currently available for tool point clouds.

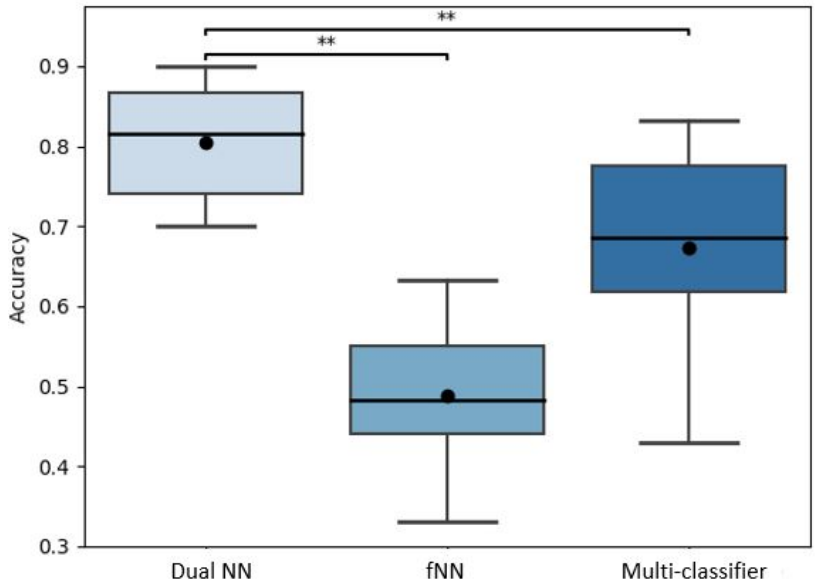


Figure 5.4: Plot showing the accuracy of our shape scoring approach for each of the six actions, when compared to baselines.

in the last layer. We use the binary cross entropy loss, with Adam optimizer. We train the model with the same dataset used for the dual networks for shape matching except, we consider individual training samples $x_i \in N$, where for any training sample x_i, $y(x_i) = 1$ if x_i is appropriate for the action.

Our results are shown in Figure 5.4. Using only shape information, we find that our approach outperforms the baselines, with an average accuracy of 81% on the testing dataset. This shows that our network is able to generalize well to previously unseen and uniquely shaped objects, by identifying salient features that make them appropriate for a given action. Abelha et al. [10] performs reasonably well using SQs (accuracy of 67%) but fNN performs poorly. Shown in Figure 5.5, we note that it is difficult to model objects such as rakes accurately using SQs, owing to the toothed structure of these objects. But SQs perform well on more regularly shaped objects such as hammers. We also note that the overall computation time for Dual NN and fNN is on average, 1.967 s and 1.911 s respectively, whereas SQ fitting with Abelha et al. [10] takes on average 342.27 s. Thus, our approach can offer computational benefits to tool substitution and improve practical applicability in terms of computation time.

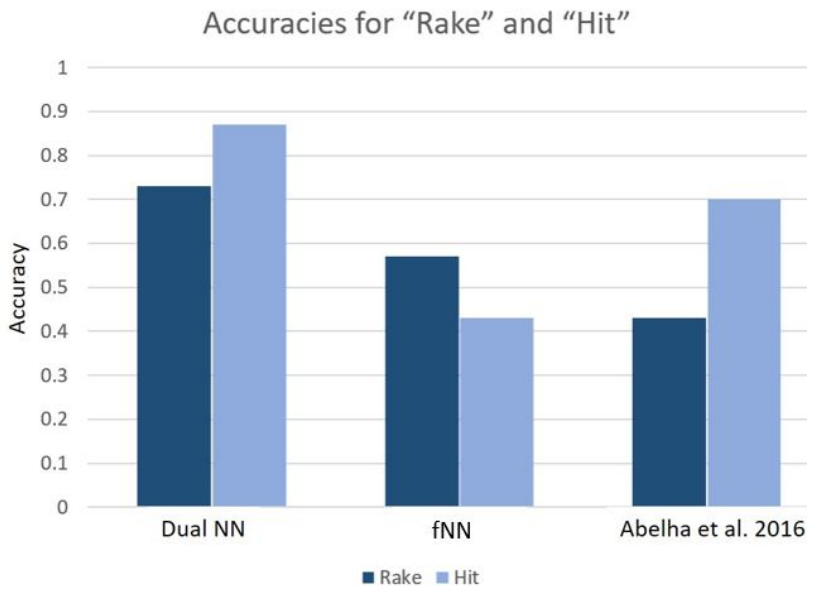


Figure 5.5: Plot showing the performance of the three approaches on the hardest (Rake) and easiest (Hit) shape scoring tasks.

5.3.2 Performance of Combined Shape and Material Reasoning

In this section, we compare the performance of tool substitution using shape scoring only, material scoring only, and combined shape and material scoring, comparing the ablations to a random ranking approach. Unlike previous tasks, which used 3D object models from 3DWarehouse, in this task we utilize real robot data. Our experimental setup is shown in Figure 5.6. 3D object scans are collected using an overhead RGBD camera, and material readings are collected by the robot using the hand-held SCiO sensor. Note that the substitution task is significantly more challenging in this real-world setting because only *partial* point clouds can be obtained from the overhead camera, and we use single spectral scans for each object, collected by a 7-DOF robot arm. Additionally, some object materials, such as stainless steel, were not included in SMM50 training data.

The 30 objects used are also shown in Figure 5.6. For validation, we created six sets of 10 objects per action (total 36 sets). Each set consisted of one “correct” substitute for the given action, and nine incorrect, which acts as our ground truth[6]. Given that our tool

[6]The correct substitute was determined by three independent evaluators (with a Cronbach’s alpha of 0.93).

Figure 5.6: Experimental setup: Shows the set of 30 objects used in experiment 3, subsection 5.3.2, along with a sample setup of the workspace with the robot shown holding the SCiO sensor.

substitution approach outputs a ranking of candidate tools, our metrics included "Hit@1", indicating the proportion of sets for which the correct tool was ranked at 1; "Average Rank", which is the average rank of the correct tool across the test sets; and "Hit@5", indicating the number of times the correct tool was ranked within the top 5 ranks of our output.

Our results are shown in Table 5.1. We found that overall, our approach combining shape and material outperformed the other conditions, with an average ranking of 2 across all the sets. In particular, we note that combining shape and material significantly improved Hit@5 (86% vs 67% for shape and 58% material only) and Hit@1 (53% vs 28% for shape and 22% material only). All three approaches performed significantly better than random ranking of the objects. While our results indicated that combining shape and material reasoning improved the performance of the tool substitution pipeline, its practical application remains a significant challenge, as indicated by the low Hit@1. We further note that using only shape information performed better than using only material information. Our results

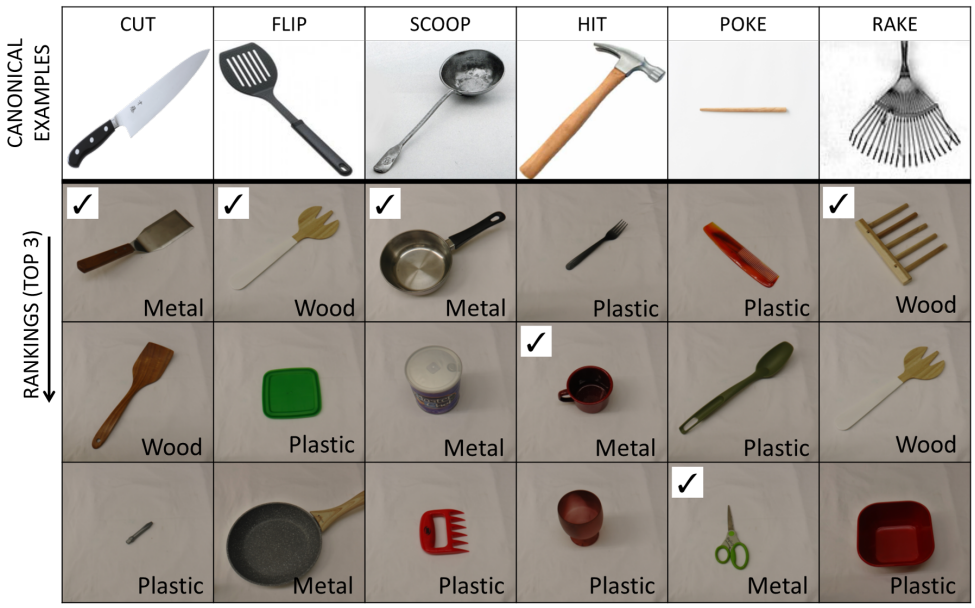


Figure 5.7: First row shows examples of some canonical tools for each action. Following rows show the ranking of objects (top 3) for some of the sets. Check marks indicate the correct outputs. The actual materials of the objects are also noted.

Table 5.1: Combined shape and material scoring performs better overall. Note that lower rank (min 1) and higher Hit@5, Hit@1 (max 1) are preferred.

	Shape Only	Material Only	Shape + Material	Random
Avg. Rank	4	5	**2**	8
Hit@1	0.28	0.22	**0.53**	0.05
Hit@5	0.67	0.58	**0.86**	0.14

using shape scoring only is interesting, since our original network was only trained on 3D models, yet it was able to generalize/transfer fairly well to real-world partial point clouds with 67% Hit@5. In contrast, we found that the spectral scans extracted by the robot posed a bigger challenge to the generalization of our material scoring system. This reflects the findings previously reported by Erickson et al. [59], and indicate that incorporating visual and haptic modalities may help improve performance.

Figure 5.7 shows some of the ranked substitutes returned by combined shape and material reasoning, for some of the test sets. The results highlight the challenges of working with

partial RGBD data and previously unseen material scans. For example, the (closed) metal can ranked as the #2 substitute tool for scooping is ranked highly, because its reflective surface resulted in a point cloud that resembled a concave bowl. Further, an incorrect material prediction for the metal mug, resulted in it being ranked as #2 substitute for performing the action of hitting.

5.4 Findings and Contributions

The two key findings of this chapter can be summarized as follows:

- Combined shape and material reasoning leads to significantly improved performance for tool substitution, when compared to material or shape only.
- Shape scoring using dual neural networks outperforms the baseline approaches for tool substitution, and is computationally faster than using SQ modeling.

The contributions of this chapter answer the research question, *"How do we enable robots to perform tool substitution?"*. We find that combined reasoning about the shape and material properties of available objects, enables the robot to effectively identify appropriate substitutes for a missing tool.

CHAPTER 6
TOOL MACGYVERING THROUGH ARBITRATION OF SUBSTITUTION AND CONSTRUCTION

How can we enable the robot to perform object improvisation?

In this chapter, we propose that behavioral arbitration using the multi-objective functions developed in the previous chapters, can enable the robot to effectively select between tool substitution and tool construction to perform object improvisation. Hence, the algorithms in this chapter take a fixed reference action, and output a constructed or substituted tool for performing the reference action. We begin by discussing open challenges in related literature. We the present our algorithmic contributions and evaluation. Finally, we summarize with our key findings.

6.1 Challenges in Behavior Arbitration

Given a set of behaviors with associated value functions, arbitration is the process of selecting one of the input behaviors based on the value functions (action selection) [24, 60, 61]. Specifically, the behavior chosen at any given instant, corresponds to the one with the highest value function. The key unaddressed challenge in behavior arbitration that this chapter seeks to tackle is:

- **Design of value functions for effective arbitration of tool substitution and construction**: Since prior work has not specifically focused on arbitration of tool substitution and tool construction, the design of appropriate value functions that can effectively arbitrate between the two behaviors is critical for tool macgyvering.

In this chapter, we present three different value functions that can be used with action selection to effectively choose between tool substitution and tool construction, and we

evaluate their relative merits. The value functions reason about the available objects using the multi-objective functions developed in the previous chapters. We present a unified tool macgyvering framework that combines tool substitution and tool construction through intelligent arbitration.

6.2 Unified Tool Macgyvering Framework for Object Improvisation

In this section, we introduce our tool macgyvering framework that unifies tool substitution and tool construction. We denote the set of all candidate objects as $O = \{o_1, o_2, ..., o_n\}$. Thus, the problem of identifying tool substitutes involves a search space of size n, where each $o_i \in O$ is a potential substitute. For tool construction, the size of the search space is combinatorial in ${}^nP_m$, assuming that we wish to construct a tool with m objects. We denote the set of all permutations of the m objects as $\mathcal{T} = \{T_1, T_2, ...\}$, where $T_i = (o_1, ..., o_m)$ is a tuple representing a specific permutation of m objects. We denote the combined space of tool substitutions and constructions as, $S = O \cup T$, where $|S| = n + {}^nP_m$. The goal of our approach is to evaluate the states in S, to identify the best tool macgyvering solution.

In order to identify the most suitable set of objects, we developed multi-objective functions, Equation 4.7 and Equation 5.3, that effectively score the fitness of the candidate objects as constructions and substitutes for performing the specified action. We showed in chapters 4 and 5 that reasoning about three features of the candidate objects within the multi-objective function, namely, ***shape*** (ϕ_{shape} or δ_{shape}), ***materials*** (ϕ_{mat}), and ***attachments*** (ϕ_{att}) in the case of tool constructions, enables the robot to effectively perform tool construction and tool substitution.

Our complete tool macgyvering framework is shown in Figure 6.1, and our complete tool macgyvering algorithm is shown in algorithm 5.

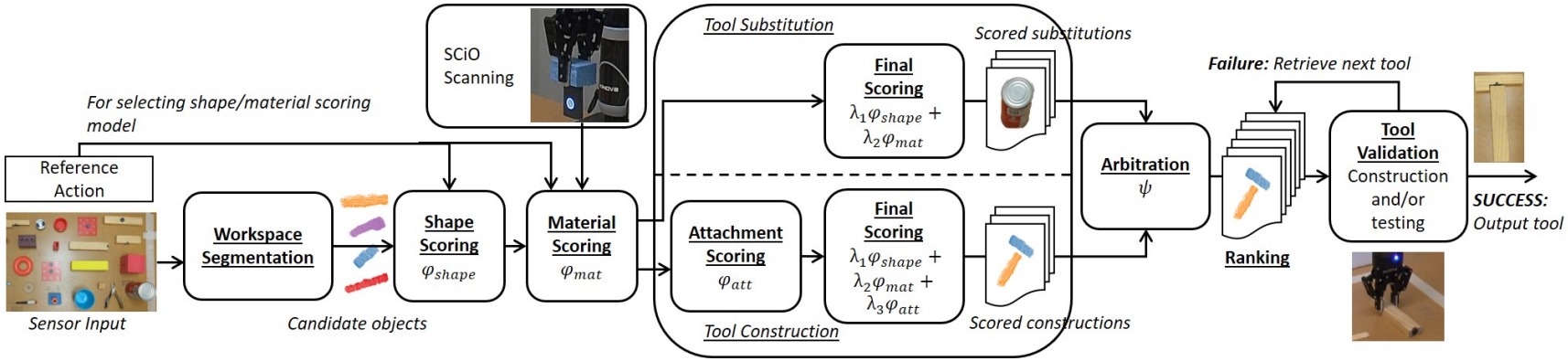


Figure 6.1: Overview of our tool macgyvering framework highlighting the different steps involved. The tool construction and substitution pipelines are followed by arbitration, to output a combined ranking of the different strategies that the robot then validates. Arbitration essentially combines substitution and construction within the framework.

Algorithm 5: Tool Macgyvering

input : $action, T = permute(O, m)$

output: T^*, Att, $Type$

1 $E = [\,]$, $Att = [\,]$, $Type = [\,]$
2 $S = O \cup T$
3 **for** $i \leftarrow 1$ **to** $|S|$ **do**
4 $\quad \phi_{shape}(s_i) = ShapeFit(s_i, action)$
5 $\quad \phi_{mat}(s_i) = MaterialFit(s_i, action)$
6 $\quad$ **if** $|s_i| > 1$ **then**
7 $\quad\quad t_{att} = AttachType(s_i)$ // *Construction with* T_i
8 $\quad\quad \phi_{att}(s_i), A_{close}(s_i) = AttachmentFit(s_i, t_{att})$
9 $\quad\quad \Phi(s_i) = \phi_{shape}(s_i) + \phi_{mat}(s_i) + \phi_{att}(s_i)$ // *Equation 4.7*
10 $\quad$ **else**
11 $\quad\quad t_{att} = \varnothing$ // *Substitution with* o_i
12 $\quad\quad A_{close}(s_i) = \varnothing$
13 $\quad\quad \Phi(s_i) = \delta_{shape}(s_i) + \phi_{mat}(s_i)$ // *Equation 5.3*
14 $\quad$ **end**
15 $\quad E.append(\Phi(s_i))$
16 $\quad Att.append(A_{close}(s_i))$
17 $\quad Type.append(t_{att})$
18 **end**
19 $V = Arbitrate(E, S)$ // *Arbitrate based on value functions*
20 $S^* = sort(S, V)$ // *Sort* S *based on* V
21 **return** $S^*, Att, Type$

The pipeline begins with **workspace segmentation** which enables the system to identify the candidate objects in the robot's workspace. We use plane subtraction and Sample Consensus Segmentation (SAC)[1] to identify the candidate objects available to the robot using RGB-D data from a camera mounted over the table. The **shape scoring** algorithm ($ShapeFit()$, algorithm 5, line 4), evaluates the visual fitness of the candidate objects and assigns a corresponding shape score (ϕ_{shape} or δ_{shape}, from Equation 4.3 and Equation 5.1 respectively). Following shape scoring, the **material scoring** algorithm ($MaterialFit()$,

[1]The implementation was provided by the PCL library

algorithm 5, line 5), evaluates the material fitness of the candidate objects, and assigns a corresponding material score (ϕ_{mat}, Equation 4.6). The shape and material scores are combined for tool substitution, in a final objective function Φ^{subs} (Equation 5.3). For tool construction, the scores discussed above do not indicate whether the objects can be attached. Hence, our **attachment scoring** algorithm ($AttachmentFit()$, algorithm 4), evaluates whether the candidate objects can be attached. The algorithm outputs an attachment score, which is combined with the shape and material scores to compute the final objective function Φ^{cons} (Equation 4.7). The final objectives are then used for computing value functions for **arbitration**. Arbitration uses the value functions to generate a combined **ranking** of the tool substitutes and constructions (ranked from highest to lowest values). Finally, the robot **validates** each construction or substitute for their task suitability, by applying the reference action with the object. In the case of construction, the robot first constructs the tool, then validates it by applying the reference action on the tool. In this work, we assume that the robot can observe whether the tool succeeded, and that the action trajectory for using the constructed or substitute tool is pre-specified. Alternatively, the action trajectory could be learned from demonstrations [75], including, if necessary, adapting the original action to fit the dimensions of the new tool [76, 77]. If the object fails at performing the action or cannot be constructed, the robot iterates through the ranks until a solution is found.

6.3 Arbitration of Tool Substitution and Tool Construction

Arbitration combines tool substitution and tool construction within our tool macgyvering framework, and in this section we present three different arbitration strategies for deciding between the two behaviors. Inspired by existing research in behavioral robotics, each behavior (substitution or construction) is associated with a value function, Ψ, that dictates the behavior chosen at a given instant [24, 60, 61]. The value functions in our work, account for the overall fitness of the substitutes and constructions for performing the reference action. We generate a combined ranking of the strategies (highest to lowest value) that the

robot iterates through, validating each strategy until a solution is found. Our set of states $S = O \cup T$, represents the union of the set of all individual objects o_i, and the set of all permutations of m objects T_i, for tool construction. We now introduce the three different value functions for arbitration, that uses the multi-objective functions introduced in the previous chapters, to compute values over the states $s_i \in S$. The final score for tool construction is described by Equation 4.7, and the final score for tool substitution is described by Equation 5.3. We repeat them for convenience here:

$$\Phi^{subs}(o_i) = \delta_{shape}(o_i) + \phi_{mat}(o_i)$$
$$\Phi^{cons}(T_i) = \phi_{shape}(T_i) + \phi_{mat}(T_i) + \phi_{att}(T_i)$$

For the arbitration strategies, first we present a **rule-based approach** that assigns a fixed value to constructions, as described below:

$$\Psi_{rule}(s_i) = \begin{cases} 10, & \text{if } |s_i| = 1, \Phi^{subs}(s_i) > 1.0 \\ 0, & \text{if } |s_i| > 1, \Phi^{cons}(s_i) > 1.0 \\ -\infty, & \text{otherwise} \end{cases} \tag{6.1}$$

Where, $|s_i|$ denotes the cardinality of $s_i \in S$, to indicate whether a single object is being evaluated (substitute, o_i) or a combination of objects (construction, T_i). This approach prefers substitutions over constructions, provided the substitutions have a higher score than a threshold. We empirically set our threshold to 1.0. A fixed value is also assigned to constructions that exceed the threshold in terms of the construction objective.

Second, we present a **direct comparison** approach that directly compares the multi-

objective functions, and assigns values to states in S as:

$$\Psi_{dir}(s_i) = \begin{cases} \Phi^{subs}(s_i), & \text{if } |s_i| = 1 \\ \Phi^{cons}(s_i), & \text{if } |s_i| > 1 \end{cases} \tag{6.2}$$

Note that the tool construction objective $\Phi^{cons}(s_i)$ automatically assigns a cost associated with attachments, namely the attachment score ϕ_{att}, and penalizes constructions over substitutions. Here, tool construction uses the part-based shape scoring approach from chapter 4 (Shown in Equation 4.3).

Third, we present a **substitution-based** approach that uses the joint shape scoring methodology described in subsection 5.2.4 for performing tool constructions (shown in Equation 5.4). This approach effectively treats the constructions as substitute objects. Hence, the values for the states in S are assigned as follows:

$$\Psi_{subs}(s_i) = \begin{cases} \Phi^{subs}(s_i), & \text{if } |s_i| = 1 \\ \delta_{shape}(s_i) + \phi_{mat}(s_i) + \phi_{att}(s_i), & \text{if } |s_i| > 1 \end{cases} \tag{6.3}$$

Here, the equation for tool construction combines the joint shape score with material and attachment scores, with uniform weighting. This enables the tool constructions and substitutions to be compared directly in terms of the shape scoring objective.

6.3.1 Evaluation of Arbitration Strategies

In this section, we evaluate the performances of the three different arbitration strategies. We validated our strategies on six different actions: 'hit', 'scoop', 'flip', 'screw', 'rake' and 'squeegee'. We created five different sets of 10 objects per action, for a total of 30 (5×6) different test cases. In each set, we included one "correct" substitute object, and one "correct" constructed object (referred to as substitution-construction pair), both of which are

Figure 6.2: The 30 test objects used for tool substitution experiments.

Figure 6.3: The 58 test objects used for tool construction experiments.

capable of performing the action, and the remainder of the objects were randomly chosen incorrect candidates. The "correct" substitutes and constructions for each pair were selected from the substitution test set (Figure 6.2) and construction test set (Figure 6.3), used in the experiments in Chapters 4 and 5. In each of the 30 test cases, we asked three independent evaluators (with Cronbach's alpha of 0.93), to evaluate which among the substitute-construction pair would be a better alternative for performing the reference action. The selected option is considered as the ground truth for that test case. For each object in the test set, the final scores were computed (Φ^{subs} or Φ^{cons}), and used in the value functions for arbitration. Thus, the final ranking generated by the value functions is a combined ranking of tool substitutes and constructions. We evaluated our arbitration strategies, both in the context of the overall ranking of the ground truth, and also in terms of the specific option,

Table 6.1: Chart showing the % number of times the correct option was chosen by each arbitration approach, along with other metrics. Bold highlights the best approach, and arrows indicate whether higher or lower values are preferred.

	Rule-based	Direct	Substitution based	Random Selection
% Correct ↑	20%	**83.33%**	60%	36.67%
Rank ↓	27.93	**9.86**	12.4	57.8
Rank % ↓	27.38%	**9.67%**	12.16%	56.67%
Hits@5 ↑	0.13	**0.43**	0.4	0.03

i.e., construction or substitution, chosen by each approach.

Our evaluation metrics include average rank, rank%, and hits@5. Rank% is the fraction of the actual rank over the total size of the state space i.e., $|S|$. Hits@5 denotes the number of cases where the ground truth was ranked within the top 5 ranks. Additionally, we include a metric that indicates the % times the correct option was chosen (% correct). We compute this by evaluating whether the arbitration strategy correctly chose between the substitution-construction pair, i.e., scored the ground truth (chosen by the human) better.

Our results in Table 6.1 show that direct comparison of scores outperforms the other approaches. In terms of ranking (rank, rank% and hits@5), we note that both direct and substitution-based approaches perform comparably. However, in terms of the % times the correct strategy was chosen, direct comparison (83.33%) outperformed substitution-based approach (60%). In our observations, the substitution-based approach was more likely to rank substitutes as better than constructions. However, both direct comparison and substitution-based approaches outperformed random selection (36.67%). Another observation is regarding the inferior performance of the rule-based approach (20%) compared to random selection, in terms of % correct. This is because rule-based almost consistently ranked substitutes as better than constructions, which did not always conform with the ground truth labels. However, it performed better in terms of average rank, rank% and hits@5. This is because the ground truth substitutes were ranked better consistently, result-

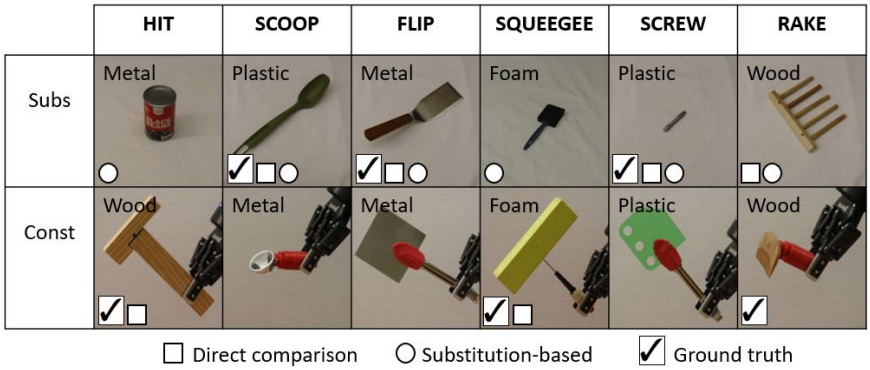


Figure 6.4: The arbitration results for direct and substitution-based approach for six substitution/construction pairs. Checkmarks indicate the human evaluated ground truth. Subs → substitution, const → construction.

ing in a better average ranking performance.

Our results in Figure 6.4 highlights some of the substitution-construction pairs in our test set, along with the selections of the direct and substitution-based strategies. Checkmarks indicate the ground truth based on human assessment. As shown in the figure, substitution-based approach was more inclined towards selecting the substitute tools over the constructions. Overall, the direct comparison approach conformed more to the ground truth assessments made by the human evaluators.

6.4 Findings and Contributions

The key findings of this chapter can be summarized as follows:

- Direct comparison of the multi-objective functions outperformed the other arbitration approaches in arbitrating between construction and substitution.

- The best design choices for our final tool macgyvering framework involve using part-based shape scoring (combined with material and attachments) for tool construction; joint shape scoring with material reasoning for tool substitution; and direct comparison for arbitration.

The final tool substitution and construction equations can be summarized as follows:

$$\Phi^{subs}(s_i) = \delta_{shape}(s_i) + \phi_{mat}(s_i)$$

$$\Phi^{cons}(s_i) = \phi_{shape}(s_i) + \phi_{mat}(s_i) + \phi_{att}(s_i)$$

Note that, the shape, material and attachment scoring take an input reference action as described in chapters 4 and 5. Given the above multi-objective functions, the final tool macgyvering equation can be described as follows, where s^* denotes the output macgyered solution, based on the arbitration strategy of direct comparison:

$$\boxed{s^* = \underset{s_i \in S}{\operatorname{argmax}}(\Phi^{subs}(s_i), \Phi^{cons}(s_i))} \tag{6.4}$$

Here, S denotes the combined space of tool substitutions and constructions, $S = O \cup T$. The equation above outputs a substituted or constructed tool for performing the input reference action. Thus, this chapter answers the question, "*how do we enable robots to perform object improvisation?"*. We find that direct comparison of the multi-objective functions for tool substitution and construction, enables the robot to perform level 1: object improvisation.

6.5 Notes on Design Choices

The objective functions described in Chapters 4, 5 and 6, can be generalized beyond the computation methods described in this dissertation. Particularly, other shape classification approaches [85, 89, 90], as well as material classification approaches [55, 58] that take input images, can be used to compute the shape and material scores respectively. In this work, we specifically developed models that were trained on tools since there is limited existing work capturing the shapes and materials of the diverse range of tools in the real world. The thresholds used in this work for the different objectives were empirically determined, although existing meta-learning approaches [11] can be useful in this direction.

CHAPTER 7

TOOL MACGYVERING IN TASK PLANNING

How can we enable robots to perform action improvisation?

In this chapter, we propose that reasoning about the given task, and the objects available for tool macgyvering, can enable the robot to perform action improvisation. In order to validate this claim, we introduce a novel approach called **Feature Guided Search (FGS)**. FGS enables efficient application of existing heuristic search algorithms in the context of task planning in order to perform tool macgyvering by accounting for physical attributes of objects (e.g., shape, material) during the search for a valid task plan. To demonstrate the computational benefits of FGS, we apply our work to the combinatorially complex problem of tool construction, although FGS can be easily extended to tool substitution as well. We begin by discussing open challenges in related literature. We then present our algorithmic contributions and evaluation. Finally, we summarize with our key findings.

7.1 Challenges in Applying Heuristic Search For Tool Construction

Heuristic search algorithms, such as A^* and enforced hill-climbing (EHC), have been successfully applied to planning problems in conjunction with heuristics such as cost-optimal landmarks [91] and fast-forward [92] respectively. However, the application of heuristic search algorithms to perform tool construction in the context of task planning can be challenging. For example, consider a task where the goal of the robot is to hang a painting on the wall. In the absence of a hammer that is required for hammering a nail to complete the task, the robot may choose to construct a replacement for the hammer using the objects available to it. This can lead to the following challenges:

- **Identifying the right object combination during the search for a task plan:** How

does the robot know which objects should be combined to construct the replacement tool? One possible solution is for the user to manually encode the correct object combination in the goal definition, and the search procedure would find it. However, it is impractical for the user to know and encode the correct object combination to use, for all the objects that the robot could possibly encounter. Alternatively, the robot can autonomously attempt every possible object combination until it finds an appropriate tool construction for completing the task. However, this would require a prohibitive number of tool construction attempts.

- **Reasoning about alternate actions:** What if the robot *cannot* construct a good replacement for a hammer using the available objects, but can instead construct a makeshift screwdriver to tighten a screw and complete the task? In this case, the task plan would also have to be adapted to appropriately use the constructed tool, i.e., "tighten" a screw with the screwdriver instead of "hammering" the nail.

In order to address these challenges, FGS combines existing planning heuristics with the multi-objective functions computed in the previous chapters, to indicate the best object combination to use for constructing a replacement tool. The chosen replacement tool then in turn guides the correct action(s) to be executed for completing the task (e.g., "tighten" vs. "hammering"). Hence, our algorithm seeks to: a) eliminate the need for the user to specify the correct object combination, thus enabling the robot to autonomously choose the right tool construction based on the available objects and the task goal, b) minimize the number of failed tool construction attempts in finding the correct solution, and c) adapt the task plan to appropriately use the constructed replacement tool (action improvisation).

7.2 Approach

In this chapter, we address the scenario in which a robot is provided with an input task that is missing some required tool. The robot must then derive a task plan that involves con-

structing an appropriate replacement tool from available objects, and use the constructed tool to accomplish the task. In this section, we begin by discussing some background details regarding heuristic search, followed by specific implementation details of FGS.

7.2.1 Heuristic Search

Heuristic search algorithms are guided by a cost function $f(s) = g(s) + h(s)$, where $g(s)$ is the best-known distance from the initial state to the state s, and $h(s)$ is a heuristic function that estimates the cost from s to the goal state. An admissible heuristic never overestimates the path cost from any state s to the goal [93, 94]. A consistent heuristic holds the additional property that, if there is a path from a state x to a state y, then $h(x) \leq d(x, y) + h(y)$, where $d(x, y)$ is the distance from x to y [93]. Most heuristic search algorithms, including A^*, operate by maintaining a priority queue of states to be expanded (the open list), sorted based on the cost function. At each step, the state with the least cost is chosen, expanded, and the successors are added to the open list. If a successor state is already visited, the search algorithm may choose to re-expand the state, *only if* the new path cost to the state is lesser than the previously found path cost [95]. The search continues until the goal state is found, or the open list becomes empty, in which case no plan is returned.

7.2.2 Feature Guided Search

We now describe the implementation of FGS[1]. For the purposes of this explanation, we present our work in the context of A^*, though our approach can be easily extended to other heuristic search algorithms as demonstrated in our experiments. Let S denote the set of states, A denote the set of actions, γ denote state transitions, s_i denote the initial state, and s_g denote the goal state. For the planning task, we consider the problem to be specified in Planning Domain Definition Language (PDDL) [27], consisting of a domain definition $\mathcal{P}_\mathcal{D} = (S, A, \gamma)$, and a problem/task definition $\mathcal{P}_\mathcal{T} = (\mathcal{P}_\mathcal{D}, s_i, s_g)$. Further, we

[1]All source code including problem and domain definitions, are publicly available at https://github.com/Lnair1993/Tool_Macgyvering

```
(:action join-spatula
    :parameters (?x - cons-part ?y - cons-part)
    :precondition (isDiff ?x ?y)
    :effect (haveSpatula)
)

(:action join-ladle
    :parameters (?x - cons-part ?y - cons-part)
    :precondition (isDiff ?x ?y)
    :effect (haveScoop)
)
```

Figure 7.1: Examples of the "join" action in PDDL for construction of spatula and ladle.

use O to denote a set of n objects in the environment available for tool construction, $O = \{o_1, o_2, ...o_n\}$. Note that, our problem formulation in this chapter converges on the guiding problem formulation of this dissertation presented in Chapter 2, namely:

Given a set of n candidate object point clouds $O = \{o_1, o_2, ...o_n\}$, and a planning task that is defined by a domain description $\mathcal{P}_\mathcal{D}$ and a problem description $\mathcal{P}_\mathcal{T}$, how can we enable the robot to perform tool macyvering for accomplishing the task goal s_g?

Since our work focuses on tools, we assume that some action(s) in A are parameterized by a set of object(s) $O_a \subseteq O$, that are used to perform the action. Specifically for tool construction, we explicitly define an action "join(O_a)", where $O_a = \{o_1, o_2, ...o_m\}, m \leq n$, parameterized by objects that can be joined to construct a tool for completing the task. For example, the action "join-hammer(O_a)" allows the robot to construct a hammer using the objects O_a that parameterize the action (Shown in Figure 7.1). For actions that are not parameterized by any object, $O_a = \emptyset$. Our approach seeks to assign a "feature score" to the objects in O_a, indicating their fitness for performing the action a. Thus, given different sets of objects O_a that are valid parameterizations of a, the feature score can help guide the search to generate task plans that involve using the objects that are most appropriate for performing the action. In the context of tool construction, the feature score guides the search to generate task plans that involve joining the most appropriate objects for constructing the replacement tool, given the objects available in the environment. Feature scoring can also potentially reject objects that are unfit for tool construction.

Algorithm 6: Feature Guided A^* Search

1 **Function** `Search` ($\mathcal{P}_\mathcal{D}, \mathcal{P}_\mathcal{T}$, *trust=true*) **:**
2 s_i, s_g = extractStates($\mathcal{P}_\mathcal{T}$)
3 A = extractActions($\mathcal{P}_\mathcal{D}$)
4 O = extractObjects()
5 $O_{reject} = [\,]$
6 openList $= [\,]$
7 setPathCost($s_i, 0$) *// Set initial state's g(s) and f(s) to 0*
8 openList.`add`($s_i, 0$)
9 **while** *OpenList* ***not*** *empty* **do**
10 currState = $argmin_s(f(s))\ \forall\ s \in$ openList
11 openList.`pop`(currState)
12 **if** *currState* = s_g **then**
13 **return** extractPlan(s_g, s_i)
14 nextStates = getNext(currState, A)
15 **for** $(s, a, O_a) \in$ *nextStates* **do**
16 $g(s)$ = computePathCost(s, currState) *// Get current path cost*
17 $c(s)$ = getPathCost(s) *// Get previous best known path cost*
18 **if** $g(s) \geq c(s)$ **then**
19 **continue**
20 **else**
21 setPathCost($s, g(s)$) *// Update lower costs as new paths are found*
22 **end**
23 $h(s)$ = computeHeuristic(s)
24 $\phi(s)$ = **featureScore**($s, a, O_a, trust$) *// Compute the feature score: Algorithm 2*
25 **if** $\phi(s) = -\infty$ **then**
26 O_{reject}.`add`(O_a, a) *// Track rejected combinations*
27 $f(s) = g(s) + h(s) - \phi(s)$
28 **if** $f(s) = \infty$ **then**
29 **continue**
30 openList.`add`($s, f(s)$)
31 **end**
32 **end**
33 **if** O_{reject} *not* $\emptyset$ **then**
34 `Search` ($\mathcal{P}_\mathcal{D}, \mathcal{P}_\mathcal{T}, trust = false$) *// Re-attempt without trusting all sensors*
35 **return** $\emptyset$ *// No plan found*

Our approach is presented in algorithm 6. The search algorithm extracts information regarding the initial state s_i and goal state s_g from the task definition (Line 2). The set of actions A is extracted from the domain definition $\mathcal{P}_\mathcal{D}$ (Line 3). The agent extracts the

objects in its environment from an RGB-D observation of the scene through point cloud segmentation and clustering (Line 4). We initialize the open list ($openList$) as a priority queue with the initial state s_i and cost of 0 (Lines 6-8). Lines 9-32 proceed according to the standard A^* search algorithm, except for the computation of the feature score in Line 24. While the open list is not empty, we select the state with the lowest cost function (Line 10,11). If the goal is found, the plan is extracted (Lines 12-13), otherwise the successor states are generated (Line 14). For each successor state s, the algorithm computes the path cost $g(s)$ from the current state $currState$ to s (Line 16). The algorithm then retrieves the best known path cost $c(s)$ for the state from its previous encounters (Line 17). If the state was not previously seen, $c(s) = \infty$. In Lines 18-22, the algorithm compares the best known path cost to the current path cost, and updates the best known path cost if $g(s) < c(s)$. The algorithm then computes the heuristic $h(s)$ (Line 23), and the feature score $\phi(s)$ (Line 24). The algorithm also maintains a list of object combinations that were rejected by feature scoring (i.e., assigned a score of $-\infty$), in O_{reject} (Line 26). The final cost is computed as $f(s) = g(s) + h(s) - \phi(s)$ (Line 27; We expand more on our choice of cost function in subsection 7.2.5). If $f(s) \neq \infty$, then the state is added to the open list, prioritized by the cost. The search continues until a plan is found, or exits if the open list becomes empty. If no plan was found, the search is reattempted (Line 34) by modifying the feature score computation (described in subsection 7.2.3). If all search attempts fail, the planner returns a failure with no plan found. In the following section we discuss the computation of the feature score in detail.

7.2.3 Feature Score Computation

In this section, we describe the computation of the feature score for a given set of objects O_a that parameterize an action a. The feature score is computed using the multi-objective functions developed in the previous chapters (Chapter 4 and Chapter 5). Specifically in this chapter, the feature score computation focuses on the problem of tool construction in-

troduced in Chapter 4. However, FGS can be extended to tool substitution, by computing the feature scores based on the multi-objective functions introduced in Chapter 5. The proposed multi-objective function from Chapter 4 included three considerations: a) shape fitness of the objects for performing the action, b) material fitness of the objects for performing the action, and c) evaluating whether the objects in O_a can be attached to construct the tool.

The calculation of each of the three metrics above relies on real-world sensing, which can be noisy. This can result in false negative predictions, that eliminate potentially valid object combinations from consideration. In particular, our results in Chapter 4 has shown that false negatives in material and attachment predictions have caused $\approx 4\%$ of tool constructions to fail [21]. To address the problem of false negatives in material and attachment predictions, we introduce the notion of "sensor trust" in this work. Prior work that has looked at accounting for sensor trust has introduced the notion of "trust weighting" to use continuous values to appropriately weigh the sensor inputs [96]. In contrast, the sensor trust parameter in our work is a *binary value* that determines whether the material and attachment predictions should be believed by the robot and included in the feature score computation. This is because material and attachment scores are hard constraints and not continuous, i.e., they are $-\infty$ for objects that are not suited for tool construction (we describe this further in later sections). Hence, a continuous weighting on the material and attachment scores is not appropriate for our work.

Our feature score computation approach is described in algorithm 7. For actions that are not parameterized by objects, the approach returns 0 (Lines 2-3). If the trust parameter is set to $true$, the feature score computation incorporates shape, material, and attachment predictions. (Lines 5-12 of algorithm 7; subsubsection 7.2.4 for details). If the trust parameter is set to $false$, the feature score computation only includes shape scoring (Lines 14-19 of algorithm 7; subsubsection 7.2.4 for details). Thus, we describe two modes of feature score computation that is influenced by the sensor trust parameter. In the following sections, we

Algorithm 7: Feature Score Computation

Function `FeatureScore`$(s, a, O_a, trust = true)$**:**
  if O_a ***is empty*** **then**
    return 0
  if *trust* **then**
    if *canAttach*(O_a, a) **then**
      $\phi^s_{shape}(O_a) = \text{ShapeFit}(O_a, a)$ // *Sensors are fully trusted - subsubsection 7.2.4*
      $\phi^s_{mat}(O_a) = \text{MaterialFit}(O_a, a)$
      $\phi(s) = \lambda_1 * \phi^s_{shape}(O_a) + \lambda_2 * \phi^s_{mat}(O_a)$ // *The weighted sum is assigned to s*
      return $\phi(s)$
    else
      return $-\infty$
    end
  else
    if $(O_a, a) \in O_{reject}$ **then**
      $\phi^s_{shape}(O_a) = \text{ShapeFit}(O_a, a)$ // *Not fully trust sensors - subsubsection 7.2.4*
      return $\phi^s_{shape}(O_a)$ // *Evaluate objects that were previously rejected*
    else
      return $-\infty$
    end
  end

briefly describe the computation of shape, material and attachment predictions.

Shape Scoring

Shape scoring seeks to predict the shape fitness of the objects in O_a for performing the action a. This is indicated by the $ShapeFit()$ function in algorithm 7. In this chapter, we consider tools to have action parts and grasp parts[2]. Thus, $m = 2$ and the set of objects O_a consists of two objects, i.e., $|O_a| = 2$. Further, the ordering of objects in O_a indicates the

[2]As in prior work, this covers the vast majority of tools [74, 11].

Table 7.1: Table indicating appropriate materials for action parts of different tools

Tool	**Material (Action part)**
Hammer	Metal, Wood
Screwdriver	Plastic, Metal
Ladle	Plastic, Wood, Metal
Spatula	Plastic, Wood, Metal
Rake	Plastic, Wood, Metal
Squeegee	Foam

correspondence of the objects to the action and grasp parts.

In order to perform shape scoring, we utilize the approach described in section 4.5, Chapter 4. We summarize the equation as follows (from Equation 4.3), using the notations described in this chapter. Given a set of objects O_a to be used for constructing the tool, let $\mathcal{K}$ denote the set of objects in O_a that are candidates for the action parts of the final tool, and let $O_a - \mathcal{K}$ be the set of candidate grasp parts. Then the shape score $\phi^s_{shape}(O_a)$ is computed by using the trained networks as before:

$$\phi^s_{shape}(O_a) = \prod_{o_i \in \mathcal{K}} p(action|o_i) \prod_{o_i \in O_a - \mathcal{K}} p(handle|o_i) \tag{7.1}$$

Where, p is the prediction confidence of the corresponding network. Thus, we combine prediction confidences for all action parts and grasp parts. For example, for the action "join-hammer(O_a)" where O_a consists of two objects (o_1, o_2), the shape score $\phi^s_{shape}(O_a) = p(hammer\_head|o_1) * p(handle|o_2)$.

Material Scoring

Material scoring seeks to predict the material fitness of the objects in O_a for performing the action a. This is indicated by the $MaterialFit()$ function in algorithm 7. In this chapter, we specifically consider the material properties of the action parts of the tool since the action parts are more critical to performing the action with the tool [23]. Further, we also

assume that the materials that are appropriate for different tools is provided a-priori, e.g., hammer heads are made of wood or metal (Shown in Table 7.1).

For material scoring, we seek to train models that can predict whether an input material is suited for performing a specific action. In contrast to the binary classification approach adopted for material scoring in section 4.7 of Chapter 4, we frame the material scoring in this chapter as a *multi-classification* problem in order to demonstrate the flexibility of the proposed multi-objective functions. As in the previous chapters, we represent the material properties of the object using spectral readings. For extracting the spectral readings, the robot uses the SCiO sensor to measure the reflected intensities of different wavelengths, in order to profile and classify object materials. The spectrometer generates a 331-D real-valued vector of spectral intensities. Then, given a dataset of SCiO measurements from an assortment of objects (we use the SMM50 dataset [59] used in Chapter 4), we train a model through supervised learning to output a class label indicating the material of the object. Our model architecture uses four hidden layers of 64, 64, 32, 32 units each, with leaky ReLU activation, and a dropout of 0.25 between each hidden layer. Our final layer uses softmax activation with categorical cross-entropy loss.

For the material score prediction, given the spectral readings for the action parts in O_a denoted by $\mathcal{K}$, we map the predicted class label to values in Table 7.1 to compute the material score using the prediction confidence of the model. Let $\mathcal{M}(a)$ denote the set of appropriate materials for performing an action a. Then the material score is computed as:

$$\phi^s_{mat}(O_a) = \begin{cases} z = \prod\limits_{o_i \in \mathcal{K}} \max\limits_{c_i \in \mathcal{M}(a)} p(c_i|o_i), & \text{if } z \geq \text{t} \\ -\infty, & \text{otherwise} \end{cases} \tag{7.2}$$

Where, p is the prediction confidence of the network regarding the class c_i. We compute the max prediction confidence across all the appropriate classes $c_i \in \mathcal{M}(a)$, and their product over the action parts in $\mathcal{K}$. For example, for the action "join-hammer(O_a)", where O_a con-

sists of two objects (o_1, o_2), the material score $\phi^s_{mat}(O_a) = \max(p(metal|o_1), p(wood|o_1))$. If the max value exceeds some threshold[3] denoted by t, then the corresponding value is returned. Otherwise, the model returns $-\infty$. Hence, note that material prediction acts as a hard constraint, by directly eliminating any objects that are made of inappropriate materials, thus reducing the potential search effort.

Attachment Prediction

Given a set of objects, we seek to predict whether the objects can be attached to construct a tool. This is indicated by the $canAttach()$ function in algorithm 7. In order to attach the objects, we consider the three attachments introduced in subsection 4.5.2 of Chapter 4, namely, *pierce attachment* (piercing one object with another, e.g., foam pierced with a screwdriver), *grasp attachment* (grasping one object with another, e.g., a coin grasped with pliers), and *magnetic attachment* (attaching objects via magnets on them). For magnetic attachments, we manually specify whether magnets are present on the objects, enabling them to be attached. For pierce and grasp attachment, we check whether the attachments are possible as described below. If no attachments are possible for the given set of objects, the feature score returns $-\infty$, indicating that the objects are not a viable combination. Note that, in contrast to predicting an attachment score as described in subsection 4.5.2 (Equation 4.4), we seek to predict a *binary label* indicating whether attachments are possible. Thus, the search eliminates objects that cannot be attached to reduce computational effort.

- **Pierce attachment:** Similar to material reasoning, we use the SCiO sensor to reason about material pierceability. We assume homogeneity of materials, i.e., if an object is pierceable, it is uniformly pierceable throughout the object. We train a neural network to output a binary label indicating pierceability of the input spectral reading [21]. If the model outputs 0, the objects cannot be attached via piercing.

- **Grasp attachment:** To predict grasp attachment, we model the grasping tool (pliers

[3]We empirically determined a threshold of 0.6 to work well

or tongs) as an extended robot gripper. This allows the use of existing robot grasp sampling approaches [80, 81, 82], for computing locations where a given object can be grasped. In particular, we use the approach discussed by ten Pas et al., that outputs a set of grasp locations given the input parameters reflecting the attributes of the pliers or tongs used for grasping [80]. If the approach could not sample any grasp locations, the objects cannot be attached via grasping.

7.2.4 Incorporating the Sensor Trust Parameter

In this section, we describe how the sensor trust parameter (Line 4, algorithm 7) is incorporated to compute the feature score in two ways. The first approach includes trusting the shape, material, and attachment predictions of the models described above. The second approach allows the robot to deal with possible false negatives in material and attachment predictions, by only incorporating the shape score into the feature score computation.

Fully trust sensors

In the case that the robot fully trusts the material and attachment predictions, the trust parameter is set to $true$ (Line 4, algorithm 7). The final feature score is then computed as a weighted sum of the shape and material scores, if the objects can be attached (algorithm 7, Lines 5-8). We found uniform weights of $\lambda_1 = 1, \lambda_2 = 1$, to work well for tool constructions. If the objects cannot be attached, then $\phi(s) = -\infty$, indicating that the objects in O_a do not form a valid combination. Otherwise, using Equation 7.1 and Equation 7.2:

$$score(s, O_a) = \lambda_1 * \phi^s_{shape}(O_a) + \lambda_2 * \phi^s_{mat}(O_a) \tag{7.3}$$

Since material and attachment predictions are hard constraints, certain object combinations can be assigned a score of $-\infty$, indicating that the robot does not attempt these constructions. However, this can lead to cases of false negatives where the robot is unable to find

the correct construction due to incorrect material or attachment predictions. In these cases, the algorithm tracks the rejected object combinations in O_{reject} (algorithm 6, Line 26), and repeats the search as described below, by setting trust to $false$ (algorithm 6, Lines 33-34).

Not fully trust sensors

In case of false negatives, the robot can choose to eliminate the hard constraints of material and attachment prediction from the feature score computation, thus allowing the robot to explore the initially rejected object combinations by using only the shape score. This is achieved by setting the trust flag to $false$ in our implementation (Lines 14-15, algorithm 7). In this case, we attempt to re-plan using the feature score as:

$$\phi(s) = \begin{cases} \phi^s_{shape}(O_a), & \text{if } O_a \subseteq O_{reject} \\ -\infty, & \text{otherwise} \end{cases} \tag{7.4}$$

Here, O_{reject} indicates the set of objects that were initially rejected by the material and/or the attachment predictions. Since, shape score is a soft constraint, i.e., it does not eliminate any object combinations completely, we use the shape score to guide the search in case of the rejected objects. In the worst case, this causes the robot to explore all ${}^nP_m$ permutations of objects. However, as shown in our results, shape score can serve as a useful guide for improving tool construction performance in practice, when compared to naively exploring all possible object combinations. The final feature score, influenced by attachments and the trust parameter, can be summarized as follows from Equation 7.3 and Equation 7.4:

$$\phi(s) = \begin{cases} score(s, O_a), & \text{if attachable \& trust} \\ \phi^s_{shape}(O_a), & \text{if } \textbf{not} \text{ trust \& } O_a \subseteq O_{reject} \\ -\infty, & \text{otherwise} \end{cases}$$

7.2.5 Final cost computation

Once the feature score is computed, the final cost function is computed as $f(s) = g(s) + h(s) - \lambda * \phi(s)$. Interestingly, we found that $\lambda = 1$, thus $f(s) = g(s) + h(s) - \phi(s)$, performs very well with the choice of search algorithms and heuristics in this work for the problem of tool construction. In this case, the higher the feature score $\phi(s)$, the lower the cost $f(s)$, in turn guiding the search to choose nodes with higher feature score (lower $f(s)$ values). Additionally, the values of the feature score are within the range $0 \leq \phi(s) \leq 2$. Since we use existing planning heuristics that have been shown to work well, and the task plans generated have $\gg 2$ steps involved, $g(s)+h(s) \gg 2$ and thus, $f(s) > 0$. Thus, $\lambda = 1$ works well for the problems described in this work. Moreover, multiplying $\phi(s)$ with $h(s)$ would cause A^* to behave as weighted A^* for $1 < \phi(s) \leq 2$. Similarly, adding $\phi(s)$ to the cost function could potentially affect admissibility. However, this presents an interesting research question for our future work in terms of an in-depth analysis of the choice of heuristic and feature score values, and its influence on the guarantees of the search.

Once a task plan is successfully found, the robot can proceed with executing the task plan and joining the parts indicated by O_a as described in Chapter 4, to construct the required tool for completing the task. If the tool could not be successfully constructed or used, the plan execution is said to have failed, and the robot re-plans to generate a new task plan with a different object combination, since the algorithm tracks the attempted object combinations. This process continues until all possible object combinations are exhausted, or the task is successfully completed.

7.3 Experimental Validation and Results

In this section, we describe our experimental setup and present our results alongside each evaluation. We validate our approach on three diverse types of tasks involving tool construction in a household domain, namely, wood-working, cooking, and cleaning. For wood-

Figure 7.2: Dataset of 58 objects used for the experiments, made of different materials

working tasks, the tools to be constructed include hammer and screwdriver; for cooking tasks the tools include spatula and ladle; and lastly for cleaning tasks the tools include rake and squeegee. Each tool is constructed from two parts ($m = 2$) corresponding to the action and grasp parts of the tool. Our experiments seek to validate the following three aspects of our approach:

1. **Performance of feature guided A^* against baselines:** In order to investigate the informativeness of including feature score in heuristic search, we evaluate the feature guided A^* approach against three baselines. We also evaluate our approach in terms of the two different settings of the sensor trust parameter to investigate the benefits of introducing sensor trust.

2. **Combining feature scoring with other heuristic search algorithms:** To investigate whether feature scoring can generalize to other search approaches, we integrate feature scoring with two additional heuristic search algorithms. Specifically, we present results combining feature scoring with weighted A^* and enforced hill-climbing with the fast-forward heuristic [92].

3. **Adaptability of task plans to objects in the robot's environment:** We evaluate whether the robot can adapt its task plans to appropriately use the constructed tool, as the objects available to the robot for tool construction are modified. This measures

whether the robot can flexibly generate task plans in response to the objects in the environment, through action improvisation.

For all our experiments, we use a test set consisting of 58 previously unseen candidate objects for tool construction (shown in Figure 7.2). These objects consist of metal (11/58), wood (12/58), plastic (19/58), paper (2/58) and foam (14/58) objects. Only the foam and paper objects are pierceable. Prior to planning, the robot scans the materials of the objects for material scoring and attachment predictions. For our results, we evaluate the statistical significance where it is applicable, using repeated measures ANOVA with post-hoc Tukey's test. We discuss each experiment in more detail below, along with the results for each.

7.3.1 Performance of Feature Guided A*

In this section, we evaluate the performance of feature guided A^* against three baselines: i) standard A^*, where $f(s) = g(s) + h(s)$, ii) feature guided uniform cost search, where $f(s) = g(s) + 2.0 - \phi(s)$, and iii) standard uniform cost search, where $f(s) = g(s)$. In ii), we use $2.0 - \phi(s)$ to add a positive value to $g(s)$ since, $0 \leq \phi(s) \leq 2$. As a heuristic with A^*, we use the cost optimal landmark heuristic [91]. We also vary the sensor trust parameter, and present results for the two cases where the robot is not allowed to change the trust parameter (trust always set to $true$, i.e., lines 33-34 of algorithm 6 not executed), and for the case where the robot is allowed to change it to $false$ when no plan is found.

For the evaluation, we create six different tasks, two tasks each for wood-working, cooking and cleaning. Each task requires the construction of *one* specific tool for its completion, e.g., one of the tasks in wood-working requires construction of a hammer, and the other requires construction of a screwdriver. For each task we created 10 test cases, where each test case consisted of 10 objects chosen from the 58 in Figure 7.2, that could potentially be combined to construct the required tool. We report the average results across the test cases for each task type (total 10×2 cases per task type with 10 candidate objects per case). We create each test set by choosing a random set of objects, ensuring that only one

Table 7.2: Table comparing feature guided A^* ("FS+H") with baselines. The other notations: "H" - standard A^*; "FS" - feature guided uniform cost search; "UCS" - standard uniform cost search. This table reports the *average* number of failed attempts per task, across test cases where tool construction was successful. Note that the max number of failed attempts possible is 89 (brute force).

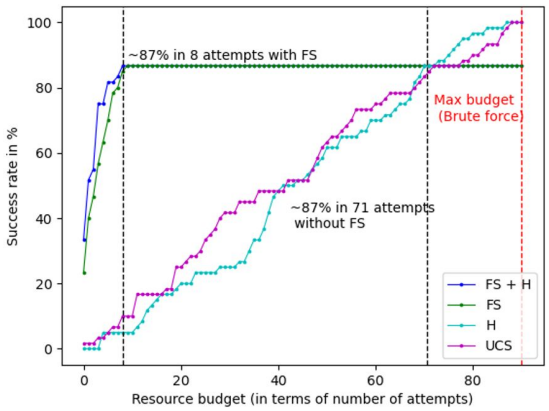
	Cleaning				**Cooking**				**Wood-working**			
	FS+H	H	FS	UCS	**FS+H**	H	FS	UCS	**FS+H**	H	FS	UCS
# Nodes	**5187**	**5187**	9061	9061	**329**	604	36237	36213	7264	**6936**	28606	28734
# Failed Attempts	**2**	46	3	49	**3**	48	4	40	**2**	45	**2**	37

"correct" combination of objects exists per set. The correct combinations are determined based on human assessment of the objects. For each task, we instantiate the corresponding domain and problem definitions in PDDL[4].

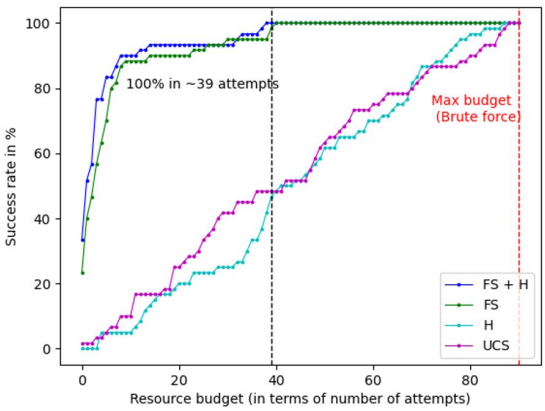
The metrics used in this experiment include i) the *number of nodes expanded* during search as a measure of computational resources consumed, ii) the *number of failed construction attempts* before a working tool was found (also referred to as "attempts" in this paper), and iii) the *success rate* indicating the number of times the robot successfully found a working tool. Ideally, we would like the number of nodes expanded and the number of failed construction attempts to be as low as possible. Note that the brute force number of failed construction attempts for 10 objects is 89, since there are ${}^{10}P_2$ possible object permutations for $m = 2$, with 89 incorrect possibilities. Ideally, we would like the number of failed construction attempts to be 0. The success rate should be as high as possible, ideally equal to 100%.

Table 7.2 shows the performance of feature guided A^* (where $f(s) = g(s)+h(s)-\phi(s)$, denoted by "FS+H") compared to the different baselines: "H" denotes standard A^* (where $f(s) = g(s) + h(s)$), "FS" denotes feature guided uniform cost search (where $f(s) = g(s) + 2.0 - \phi(s)$), and "UCS" denotes standard uniform cost search (where $f(s) = g(s)$).

[4]In the planning problem definition, the objects are instantiated numerically through "obj0" to "obj9", where each literal is automatically grounded to one of the 10 objects during planning time. Our planning and domain definitions are available at https://github.com/Lnair1993/Tool_Macgyvering.



(a) Graph showing the success rate compared to the number of attempts when **sensors are fully trusted**



(b) Graph showing the success rate vs. the number of attempts when **sensors are not fully trusted**

Figure 7.3: Graphs highlighting the success rates for the two different modes of feature scoring based on sensor trust parameter, in relation to the number of failed attempts. Note that X-axis highlights the *actual* number of attempts across all test cases for wood-working, cooking and cleaning put together.

The values reported per task are the average performances across the test cases where tool constructions were successful. As shown in Table 7.2, incorporating feature scoring (FS, FS+H) helps significantly reduce the number of failed construction attempts compared to the baselines without feature scoring (H, UCS), with $p < 0.001$. Since heuristics can help reduce the search effort in terms of number of nodes expanded, we see that approaches that do not use heuristics (FS and UCS) expand significantly more nodes than FS+H and H, with $p < 0.001$. Note that there is no statistically significant difference in the number of nodes expanded between H and FS+H. Thus, using feature scoring with heuristics (FS+H) yields the best performance in terms of *both* number of nodes expanded, and the number of failed construction attempts. **To summarize, these results show that feature scoring is informative to heuristic search by significantly reducing the average number of failed construction attempts to $\approx$ 2 compared to $\approx$ 46 without it (brute force number of failed attempts is 89).**

Further, in Figure 7.3a and Figure 7.3b, we plot the success rate vs. the resource budget of the robot in terms of the *permissible* number of failed attempts. That is, the robot is not

allowed to try any more than a fixed number of attempts, indicated by the resource budget. Figure 7.3a considers the case where the sensor trust parameter is always set to $true$, and Figure 7.3b considers the case when the robot is allowed to switch the trust to $false$, if a solution was not found. Note that in contrast to Table 7.2, the graphs report *actual* number of failed attempts, *across all tasks*, whereas Table 7.2 reports the *average* number of failed attempts across the test cases per task, for tool constructions that were successful. In Figure 7.3a, we see that FGS (FS+H and FS) achieves a success rate of 86.67% (52/60 constructions) within a resource budget of ≈ 8 failed attempts to do so. This indicates that 13.33% of the valid constructions were treated as false negatives by material and attachment predictions, and were completely removed from consideration (unattempted). Thus, increasing the permissible resource budget beyond 8, does not make any difference. Without feature scoring, H and UCS achieve a success rate of 87% with a budget of 71 attempts, and 100% after exploring nearly every possible construction (max resource budget of 89 failed attempts). In contrast, when the robot is allowed to switch the trust parameter, the robot uses shape scoring alone to continue guiding the search. As shown in Figure 7.3b, FGS (FS+H and FS) achieves 100% success rate within a budget of ≈ 39 attempts, since the robot does not eliminate any object combinations from consideration. The performance is also significantly better than the baselines that do not use feature scoring. This is because shape scoring guides the search through the space of object combinations based on the objects' shape fitness, compared to H and UCS that do not have any measure of the fitness of the objects for tool construction. **To summarize, feature scoring enables the robot to successfully construct tools by leveraging the sensor trust parameter, while significantly outperforming the baselines in terms of the resource budget required.**

In order to understand which tools were more challenging for feature scoring, Table 7.3 shows a tool-wise breakdown in performance for feature guided A^* for the two different sensor trust values. The notation "trust" denotes the case where sensors are fully trusted, and "$\sim$trust" denotes case where they are not fully trusted. When the sensors are fully

Table 7.3: Table showing tool-wise breakdown in performance for feature guided A^*. This table reports the *average* number of failed attempts per tool, across cases where tool construction was successful. The notation ∼trust indicates cases where sensors are *not* fully trusted. Note that max # failed attempts is 89.

	Cleaning				**Wood-working**				**Cooking**			
	Squeegee		**Rake**		**Hammer**		**Screwdriver**		**Spatula**		**Ladle**	
	trust	**∼trust**	**trust**	**∼trust**	**trust**	**∼trust**	**trust**	**∼trust**	**trust**	**∼trust**	**trust**	**∼trust**
# Failed Attempts	0	1	3	7	2	2	2	8	3	7	2	2
# Success	9/10	10/10	7/10	10/10	10/10	10/10	8/10	10/10	8/10	10/10	10/10	10/10

Figure 7.4: (Left) A sample task plan where a spatula must be constructed for a cooking task, and the planner uses the flat piece (obj4 in the problem definition), and tongs (obj5 in the problem definition). The action "join-spatula" refers to the construction of the spatula using obj4 and obj5. Similarly, (right) a squeegee is constructed from obj1 (foam block) and obj6 (screwdriver) for the cleaning task. Without tool construction (highlighted in green) the actions underlined in red would fail.

trusted, rakes were a particularly challenging test case, as indicated by the lowest success rate of 7/10. In contrast, hammers and ladles have a success rate of 10/10. The failure cases for each tool arises from incorrect material and attachment predictions. While not fully trusting the sensors (∼trust) leads to a 100% success rate (60/60 cases), using shape score alone leads to more failed construction attempts when compared to combining shape with material and attachment predictions since shape alone is less informative (e.g., for rake, ∼trust has 7 failed attempts vs. 3 failed attempts for trust).

Table 7.4: Table showing performance of feature guided Weighted A^* (wA^*) and feature guided Enforced Hill-Climbing (eHC) with the fast-forward heuristic (FF).

	Cleaning			**Cooking**			**Wood-working**		
	A^*	**w**A^*	**eHC**	A^*	**w**A^*	**eHC**	A^*	**w**A^*	**eHC**
# Nodes	5187	21	21	329	23	35	7264	25	38
# Failed Attempts	2	2	4	3	3	4	2	1	2
Plan length	20	22	22	19	19	19	11	15	15

Figure 7.4 shows sample task plans generated by the robot in cooking and cleaning tasks. In the case of cooking, the robot needed a spatula to flip the eggs, and used a flat piece (obj4) with tongs (obj5) to construct the spatula via grasp attachment. For cleaning, the robot needed a squeegee to clean the window, and used a foam block (obj1) and screwdriver (obj6) to construct the squeegee via pierce attachment. Without the constructed tools, the actions highlighted in red would fail, i.e., the "flip" action would fail without the constructed spatula. Hence, **FGS enables the robot to replace missing tools through construction. To summarize, the key findings of this experiment indicate that feature scoring is highly informative for heuristic search by reducing the number of nodes expanded by $\approx$ 82%, and the number of failed construction attempts by $\approx$ 93%, compared to the baselines. Further, allowing the robot to switch the trust parameter when a plan is not found, helps achieve a success rate of 100% within a budget of $\approx$ 39 attempts, significantly outperforming baselines that do not use feature scoring.**

7.3.2 Feature Scoring With Other Heuristic Search Algorithms

To demonstrate that feature scoring generalizes to other search approaches, in this section we present results for combining feature scoring with weighted A^* [97], and enforced hill-climbing using the fast-forward heuristic [92]. We use the same experimental setup and metrics as described in subsection 7.3.1. In addition, we also measure the output plan length to investigate the optimality of the different approaches. For weighted A^*, feature scoring

is incorporated as $f(s) = g(s) + w * (h(s) - \phi(s))$, where w indicates a weight parameter[5]. For enforced hill-climbing, the cost function is computed as $f(s) = h(s) - \phi(s)$. For both weighted A^* and enforced hill-climbing, we use the fast-forward heuristic, which has been shown to be successful for planning tasks in prior work [92].

In Table 7.4, we present the results for feature scoring combined with A^* and the cost-optimal landmark heuristic ("A^*+LM"), weighted A^* with fast-forward heuristic ("wA^*+FF"), and enforced hill-climbing with fast forward heuristic ("eHC+FF"). Compared to A^*+LM, wA^*+FF and eHC+FF reduce the computational effort (fewer nodes expanded) in return for sub-optimal solutions (longer plan lengths). This is expected of weighted A^* and enforced hill-climbing since they are inadmissible algorithms. There is no statistically significant difference between # failed construction attempts in each case. **To summarize, the key finding of this experiment is that feature scoring can be applied to other planning heuristics such as fast-forward, and other heuristic search algorithms like weighted A^* and enforced hill-climbing, to further reduce computational effort, albeit at the cost of optimality in terms of plan length.**

7.3.3 Adaptability of Task Plans

In this section we evaluate the adaptability of our FGS approach to generate task plans based on the objects available in the environment, to appropriately use the constructed tool through action improvisation. We create three tasks, one task each for wood-working, cooking and cleaning. In each of the tasks, *either* of two tools can be constructed to successfully complete the task, but there is only one ground truth depending on the objects available for construction. That is, the available objects only enable the construction of one of the two tools. Thus, the robot has to *correctly choose the tool to be constructed*. In addition, the robot must adapt the task plan to appropriately use the constructed tool. For the wood-working task either a hammer (with action "hit") or a screwdriver (with action

[5]Weight was set to 5.0

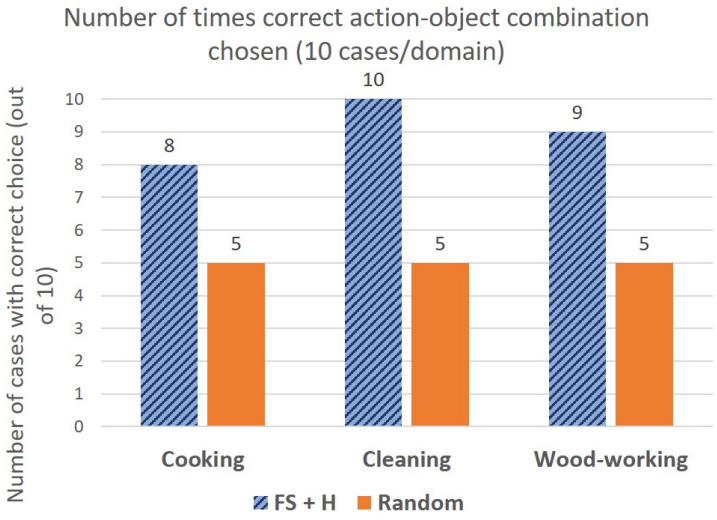


Figure 7.5: Graph highlighting the number of times the correct object combination was chosen, compared to the random selection baseline. FGS significantly outperforms random baseline ($p < 0.01$).

"tighten") can be used to attach two pieces of wood; for the cooking task either a spatula (with action "flip") or a ladle (with action "scoop") can be used to flip eggs; and for the cleaning task, either a squeegee (with action "reach") or a rake (with action "collect") can be used to collect garbage.

For the evaluation, we create three different tasks, one each in wood-working, cooking, and cleaning. For each task, either one of two tools can be used to complete the task as described above. For each task, we created 10 different test sets of random objects, similar to the experiment described in subsection 7.3.1. In each case, only one "correct" combination exists. Thus, the robot has to correctly identify which of the two tools can be constructed for accomplishing the task, given the set of objects. We evaluate the performance of feature guided A^* in each case alongside a random selection baseline to demonstrate the difficulty of the problem. The random selection baseline randomly chooses one of the two tool construction options for each task. Note that for each task, the domain and problem definitions are unchanged across the 10 test cases of objects. This indicates that the task plan adaptability does not require any manual modifications by the user, instead is the direct result of the sensor inputs received by the robot.

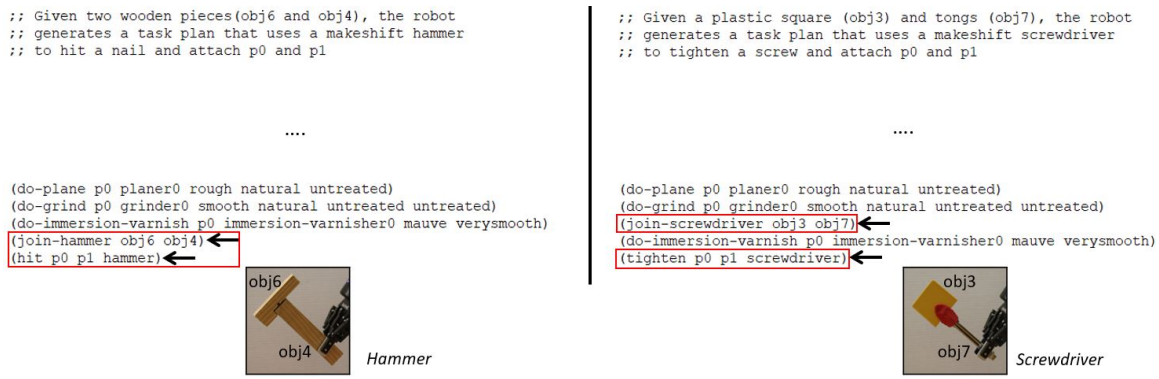


Figure 7.6: This figure shows the results for two of the test cases in wood-working, demonstrating action improvisation. The task plans are adapted based on the constructed tool (i.e., hammer or screwdriver), to either "hit" or "tighten" to attach the two pieces of wood $p0$ and $p1$. Arrows denote the parts of the task plan that are adapted.

The key metric used in this experiment includes the number of times the robot chose the correct tool to construct for each task. Thus, if the robot chose to construct a hammer, when the correct combination was to construct a screwdriver, the attempt is considered to have failed. We also present qualitative results showing some of the sample task plans and tools constructed by the robot for different sets of objects.

Figure 7.5 shows the performance of feature guided A^* compared to the random selection baseline. We see that feature guided A^* chooses the correct tool for 27/30 cases, and significantly outperforms the random selection baseline ($p < 0.01$). The failure cases in the wood-working task arise due to noisy material detection. In the case of cooking task, the noisy point clouds sensed by the RGBD camera leads to incorrect choices, e.g., the concavity of bowls was not correctly detected for some ladles.

In Figure 7.6, we show two task plans that are generated within the task of wood-working. For the same task, either a hammer or a screwdriver can be used to attach two pieces of wood $p0$ and $p1$. Depending, on the objects available in the environment, the robot chooses to construct one of the two tools and adapts the task plan to use the corresponding tool for completing the task. As shown in the left of Figure 7.6, the robot chose to construct a hammer to "hit" and attach the two pieces of wood. Whereas, shown in the right of Figure 7.6, the robot chose to construct a screwdriver to "tighten" and attach the two pieces

Figure 7.7: Collage indicating sample tool constructions output for two test cases per task. The solid and dashed brackets indicate the test set of objects provided in each case, along with the tool constructed for it. As the objects are changed, the corresponding constructed tool and action is different. Note that the problem and domain definitions are fixed for each task, and unchanged across the test cases per task.

of wood. Similar adaptations are observed for the remaining two tasks as well: "scoop" with ladles vs. "flip" with spatulas in the cooking task, and "reach" with squeegees vs. "collect" with rakes in the cleaning task. Thus, the constructed tool depends on the objects in the environment, which in turn adapts the generated task plan with alternate actions to appropriately use the constructed replacement tool. Note that both actions in each case accomplishes the same reference effect, but through the use of two different objects. Hence, the results in this section demonstrates level 2: action improvisation.

In Figure 7.7, we present some qualitative results for six different tools constructed by the robot for six of the test cases. The solid and dashed parentheses highlight the input test set. For example, given the metal bowl and metal pliers, the robot chooses to construct a

ladle (and use the "scoop" action in the task plan). In contrast, when the pliers and bowl are replaced with a plastic handle and a flat plastic piece, the robot chooses to construct a spatula instead (and use the "flip" action in the task plan). Given that the problem and domain definitions are unchanged for the two cases, this shows that the robot is able to adapt the task plan in response to the objects in the environment. **To summarize, the key finding of this experiment is that the robot is able to successfully perform action improvisation to construct and use the appropriate tool depending on the objects available for construction, with an accuracy of 90% (27/30 cases).**

7.4 Findings and Contributions

The key findings of this chapter can be summarized as follows:

- FGS significantly reduces the number of nodes expanded by $\approx 82\%$, and the number of construction attempts by $\approx 93\%$, compared to standard heuristic search baselines.
- The approach achieves a success rate of 87% within a resource budget of 8 failed attempts when sensors are fully trusted, and 100% within a resource budget of 39 failed attempts, when the sensors are not fully trusted.
- FGS enables flexible generation of task plans based on the objects in the environment, by adapting the task plan through action improvisation in order to appropriately use the constructed tool.
- Feature scoring can also be effectively combined with other heuristic search algorithms such as weighted A^* and enforced hill-climbing.

The contributions of this chapter answer the final research question of this dissertation, *"How can we enable robots to perform action improvisation?"*. We find that reasoning about the task goal, and the shape and material properties of available objects, enables robots to effectively perform action improvisation. Further, we efficiently integrate tool

macgyvering within a task planning framework to address the core problem formulation that guides this thesis (introduced in Chapter 2).

CHAPTER 8
CONCLUSIONS AND OPEN QUESTIONS

We envision a future where robots are able to innovate and engineer solutions to problems in order to assist humans more effectively. Towards achieving this long-term vision, this dissertation focused on the problem of tool macgyvering. We contributed novel algorithms and frameworks to enable robots to innovate tools in situations where the required tools for a task are missing, focusing on both tool substitution and tool construction to do so. More specifically, our thesis set out to validate the claim:
Given a task where the required tools are unavailable, a robot can efficiently perform tool macgyvering by reasoning about the task, and properties of the available objects including their shape and material.

8.1 Summary of Thesis Contributions

Toward that end, this dissertation has made the following contributions:

8.1.1 Formalization of Three Levels of Tool Macgyvering

We introduced a formalization of tool macgyvering with three levels of complexity, by leveraging the notion of affordances, and affordance equivalences. This formalization seeks to guide future research and development of tool macgyvering algorithms. The formalization also enabled us to classify the algorithms developed in this dissertation.

8.1.2 Tool Construction Using Shape, Material and Attachment Reasoning

We introduced a novel multi-objective function that reasoned about the shape, material, and attachment capabilities of available objects, to effectively rank object combinations that are

viable candidates for tool construction. We used supervised learning techniques for learning each of the three objectives. Our results on a physical robot platform demonstrated that combined reasoning about shape, material, and attachment capabilities of objects enables robots to efficiently construct a diverse set of tools.

8.1.3 Tool Substitution Using Shape and Material Reasoning

We contributed a novel tool substitution algorithm that jointly reasons about the shape and material of available objects to effectively rank the objects that are potential substitutes for a missing tool. We use supervised learning using dual neural networks to perform the shape and material reasoning. Our results demonstrated that combined reasoning about shape and material improved tool substitution performance, when compared to using either material or shape only.

8.1.4 Tool Macgyvering Through Arbitration of Substitution and Construction

We contributed three arbitration techniques for deciding between tool substitution and tool construction as the more appropriate solution for a given task and set of objects. We also contributed a novel tool macgyvering framework that integrated tool construction and tool substitution through intelligent arbitration. Our results demonstrated that out of the three arbitration techniques, "direct comparison" using the multi-objective functions developed for tool substitution and construction, outperformed the other approaches in terms of concurring with the choices made by a human. The approach was also able to effectively select between tool substitution and tool construction with high accuracy.

8.1.5 Tool Macgyvering in Task Planning

We contributed the *Feature Guided Search* (FGS) approach that enables efficient application of existing heuristic search algorithms in the context of task planning in order to perform tool construction by accounting for physical attributes of objects (e.g., shape, ma-

terial) during the search for a valid task plan. FGS combines existing planning heuristics with a feature score that is computed from the multi-objective functions developed in this thesis, in order to indicate the best object combination to use for constructing a replacement tool. Our results demonstrated that feature score is highly informative to heuristic search by significantly reducing the overall computational effort. Further FGS was able to adapt task plans to appropriately use the constructed replacement tool through action improvisation.

8.2 Open Questions

There are several insights gained from the contributions of this thesis that lead to interesting open questions for future work. We categorize the open challenges into three broad categories concerning a) the three levels of macgyvering, b) learning in tool macgyvering, and c) tool construction from deformable materials. We discuss each of them in this section.

8.2.1 Levels of Macgyvering

The contributions of this dissertation have introduced novel algorithms primarily for solving level 1 macgyvering (object improvisation) problems, with Chapter 7 introducing the Feature Guided Search approach for tackling level 2 macgyvering (action improvisation). However, several open questions remain with respect to each level as discussed below.

Level 1: Object Improvisation

The approaches in this work have focused primarily on bi-modal, i.e., visual and spectral reasoning in order to perform tool macgyvering. For tool construction, the robot reasons about three specific types of attachments for joining rigid objects, namely pierce, grasp and magnetic attachments. However, in many cases the proposed attachment modalities may not be viable options. Furthermore, some non-rigid objects, such as clay, may not retain their shape properties when manipulated. To be able to generalize to a larger set of tool constructions and more diverse objects, the robot will have to additionally reason about,

a) other forms of attachment such as gluing or taping objects together, and b) multi-modal reasoning that incorporates tactile information, particularly when dealing with malleable or soft objects.

Level 2: Action Improvisation

While the Feature Guided Search approach reasons about alternate actions that enable the robot to appropriately use the constructed tool, a key open question is action discovery. Specifically, how can the robot learn new actions that were not encoded a-priori in the domain definition of the planner? This would allow the robot to extend its capabilities beyond the predefined set of actions, to generate potentially more efficient solutions to tool macgyvering problems.

Another key question concerns the representation of the action. In this dissertation, we have focused on high-level PDDL descriptions of actions in Chapter 7, to adapt the task plans to appropriately use the constructed tool. However, the low-level motion trajectories should be adapted to appropriately use the constructed tool as well. In our thesis, we assumed that the motion trajectories for using the constructed tool is pre-specified rather than learned. Hence, an interesting extension of this work would be to learn appropriate motion trajectories for using the constructed tools.

Level 3: Sub-goal Improvisation

Level 3 macgyvering presents the most challenging set of tool macgyvering problems, that lie outside the scope of this dissertation. Tackling level 3 macgyvering requires the robot to be able to reason about alternate strategies for accomplishing a task. Specifically, the robot would have to potentially reason about, a) the task goal, and b) task plans or sequence of actions, both at the high-level task definitions, and low-level task execution. Consider an example task of lighting a room, by either switching on a lamp, or by opening the curtains. This involves two different actions, namely, "*switch on*" and "*open*", with two

different effects, namely, "*lamp is ON*" and "*curtain is OPEN*". However, at a more abstract level, both actions accomplish the task of lighting the room. This is a challenging problem that requires reasoning at the appropriate level of abstraction. A direct, and perhaps less impressive solution would be to manually encode the effects of both actions as "lighting the room". However, an open challenge is to enable the robot to autonomously reason about its capabilities at the appropriate level of abstraction to solve the task.

This leads to another key question: What is a good representation for tasks, that can enable the robot to reason at different levels of abstraction? Such a task representation would have to be hierarchical in nature, and potentially represented by graphical models to capture relationships between different sub-tasks or sub-goals. Other interesting open questions include, a) how to build such a representation, and b) how to reason over the representation in order to generate alternate strategies for accomplishing the task.

8.2.2 Learning in Tool Macgyvering

The multi-objective function developed in this dissertation used supervised learning to reason about the three different objectives of shape, material and attachment capabilities of objects. However, the relative weights associated with each objective had to be empirically determined and manually tuned. Hence, an interesting open question is, can the robot learn a good objective function through some form of exploratory learning with the use of existing knowledge bases to efficiently guide the exploration? This would involve integrating prior information regarding affordances of objects with end-to-end learning systems, in order to directly identify suitable candidates for tool macgyvering, without requiring significant parameter tuning with respect to the weights of the multi-objective function. While in our work, the empirically determined weights worked well across the tasks in our experiments, it is possible that in certain cases the weights may have to be adapted depending on the available objects and the task. Enabling the robot to do so, can greatly increase the robot's capabilities to generalize to a wider range of tools and scenarios.

8.2.3 Tool Construction From Deformable Materials

The contributions of this dissertation have focused on attaching rigid objects together to construct tools. An interesting open question is, can the robot craft novel tools from deformable materials? For instance, bending a pipe-cleaner or wire to create a hook. The capability to manipulate deformable materials for tool construction can further enhance the adaptability of robots by enabling them to create a wider range of tools. Learning to deform materials may potentially involve a trial-and-error approach, making deep reinforcement learning a suitable candidate for solving the problem. However, standard deep reinforcement learning techniques may be intractable for this problem owing to the size of the action and state space, thus requiring some form of structured knowledge, e.g., action or shape priors. Specifically, utilizing a-priori information regarding what shapes afford particular actions, or using the multi-objective functions developed in this dissertation, could potentially guide the learning process, to make the problem significantly more tractable.

8.3 Ethical Considerations

In this section, we briefly discuss the ethical considerations associated with autonomous tool macgyvering. Beyond the scope of this dissertation, autonomous tool macgyvering can greatly benefit from expert review and discussion in order to ensure that it is ethically implemented. Particularly, in order to ensure safety of the human users, it is important to enable the robot to effectively and accurately identify what objects or entities are *not* permissible for tool macgyvering. Any such entities must be excluded from the candidate set of objects. Prior to the deployment of such systems outside of a controlled laboratory setting, it is important to engage researchers with diverse perspectives to accurately assess the implications of this work. In terms of the algorithms themselves, implementing user detection systems along with careful selection of the confidence thresholds within the multi-objective functions is essential to ensure safety, and must be investigated. Further,

incorporating user advice, i.e., having the robot verify its solution with a human before proceeding with the physical tool macgyvering, can be beneficial.

8.4 Summary

Through novel algorithms and frameworks, this thesis enables robots to efficiently perform tool macgyvering to solve tasks where required tools are unavailable, thus improving the robot's adaptability to unforeseen situations. Moreover, this thesis is one of the first works to demonstrate tool macgyvering capabilities on a physical robot. By further building upon and extending the technical contributions in this thesis, we can progress towards developing the next generation of adaptive, inventive and resourceful robots that are capable of innovating solutions to problems using the resources available in their environment.

Appendices

APPENDIX A
VIDEOS AND PRESS COVERAGE

Videos demonstrating the execution of tool construction on a physical 7-DOF robot arm are available at the following links:

- **Robot Tool Macgyvering - Autonomous Tool Construction from Available Parts**: https://www.youtube.com/watch?v=_Akr8KKj9Hk
- **Autonomous Tool Construction Using Part Shape and Attachment Prediction**: https://www.youtube.com/watch?v=1XhS3Ljduts&t
- **'Tool Macgyvering': Georgia Tech Robots Learning to Construct Simple Tools**: https://www.youtube.com/watch?v=ZSObU3AFbhU&t

A.1 Press Coverage

Following are selected news articles covering our thesis work:

- **Engadget, 2019**: Georgia Tech researchers teach robots to be mechanical MacGyvers
- **Quartz, 2019**: We're teaching robots to build their own tools
- **Fortune, 2019**: Watch: Georgia Tech's Robot MacGyver Can Fashion Tools From Spare Parts
- **Georgia Tech College of Computing, 2019**: Robot First to 'MacGyver' Simple Tools by Assessing Objects' Form, Function
- **Georgia Tech College of Engineering, 2020**: The Formula of Creativity

APPENDIX B

PLANNING DOMAIN AND PROBLEM DEFINITIONS

Following are the PDDL domain and problem definitions for one cooking task from chapter 7. More examples are available at the links provided in the Codebase and Data Sections.

B.1 Domain Definition for Cooking Task

```
(define (domain cooking)
   (:requirements :typing)
   (:types container obj location cons-part spray)
   (:predicates (have ?x) (in ?x ?y) (on ?x ?y) (empty ?x) (isMixable ?x)
(isHalfCooked ?x) (isFullCooked ?x) (atLoc ?x ?y) (isPourable ?x)
(isSliceable ?x) (isSpatula ?x) (isScoop ?x) (isKnife ?x) (sprayed ?x)
(mixed ?x) (isUtensil ?x) (isSliced ?x) (isBlender ?x) (isSmoothie ?x)
(isIngredient ?x) (isAttached ?x ?y) (isGrated ?x) (haveScoop) (haveSpatula)
(isDiff ?x ?y) (isSqueegee ?x) (isGrater ?x) (isClean ?x)
(isWashed ?x) (isDirty ?x) )

   (:action flip :parameters (?x - obj ?y - container ?m - obj)
:precondition (and (isSpatula ?m) (haveSpatula) (isUtensil ?y) (in ?x ?y)
(isHalfCooked ?x))
:effect (isFullCooked ?x) )

   (:action grab :parameters (?x - obj ?l - location)
:precondition (atLoc ?x ?l)
:effect (and (have ?x) (not (atLoc ?x ?l))) )
```

```
   (:action pour :parameters (?x - obj ?y - container)
:precondition (and (isPourable ?x) (empty ?y) (have ?x))
:effect (and (in ?x ?y) (not (empty ?y)) (not (have ?x))) )

   (:action transfer :parameters (?x - obj ?y - container ?z - container)
:precondition (and (empty ?z) (in ?x ?y))
:effect (and (empty ?y) (not (empty ?z)) (in ?x ?z) (not (in ?x ?y))) )

   (:action stir :parameters (?x - obj ?y - container ?z - obj)
:precondition (and (isScoop ?z) (in ?x ?y) (haveScoop) (isMixable ?y))
:effect (mixed ?x) )

   (:action spray :parameters (?y - container ?x - spray)
:precondition (empty ?y)
:effect (sprayed ?y) )

   (:action fry :parameters (?x - obj ?y - container)
:precondition (and (mixed ?x) (sprayed ?y) (isUtensil ?y) (in ?x ?y))
:effect (and (isHalfCooked ?x) (not (sprayed ?y))) )

   (:action cut :parameters (?x - obj ?y - obj)
:precondition (and (isWashed ?x) (isKnife ?y) (isSliceable ?x) (have ?y)
(have ?x))
:effect (isSliced ?x) )

   (:action blend :parameters (?x - obj ?z - container)
```

```
:precondition (and (isSliced ?x) (isBlender ?z) (in ?x ?z))
:effect (isSmoothie ?x) )

   (:action add :parameters (?x - obj ?z - obj)
:precondition (and (isIngredient ?x) (have ?x))
:effect (in ?x ?z) )

   (:action scoop :parameters (?x - obj ?y - obj ?z - obj)
:precondition (and (isIngredient ?x) (isScoop ?z) (haveScoop) (isSmoothie
?y) (have ?x))
:effect (on ?x ?y) )

   (:action sprinkle :parameters (?x - obj ?y - obj)
:precondition (and (isGrated ?x) (isFullCooked ?y) (have ?x))
:effect (and (on ?x ?y) (not (have ?x))) )

   (:action grate :parameters (?x - obj ?y - obj)
:precondition (and (have ?x) (have ?y) (isGrater ?y))
:effect (isGrated ?x) )

   (:action wash :parameters (?x - obj)
:precondition (and (isDirty ?x))
:effect (isWashed ?x) )

   ;; Attach action
   (:action join-flip :parameters (?x - cons-part ?y - cons-part)
:precondition (isDiff ?x ?y)
```

```
:effect (haveSpatula) )

   (:action join-scoop :parameters (?x - cons-part ?y - cons-part)
:precondition (isDiff ?x ?y)
:effect (haveScoop) )

   (:action squeegee :parameters (?x - obj ?y - obj)
:precondition (and (have ?y) (isDirty ?x) (isSqueegee ?y))
:effect (isClean ?x) )
)
```

B.2 Problem Definition for Cooking Task

```
(define (problem cooking-eggs)
(:domain cooking)

   (:objects fridge cabinet table shed freezer - location eggs strawberry
spatula scoop knife icecream salt - obj ;;cheese grater squeegee pan bowl
plate0 blender glass - container ;;plate1 oil-can - spray obj0 obj1 obj2
obj3 obj4 obj5 obj6 obj7 obj8 obj9 - cons-part )

   (:init (atLoc eggs fridge) (atLoc icecream freezer) (atLoc strawberry
fridge) (atLoc salt table) (atLoc scoop cabinet) (atLoc knife cabinet)
(empty pan) (empty bowl) (empty plate0) (empty glass) (empty blender)
(isSpatula spatula) (isIngredient salt) (isIngredient icecream)
(isScoop scoop) (haveScoop) (isMixable bowl) (isPourable eggs)
(isPourable strawberry) (isUtensil pan) (isBlender blender) (isKnife knife)
(isSliceable strawberry) (isDirty strawberry)
```

```
;; Differentiate parts
(isDiff obj1 obj2) (isDiff obj1 obj3) (isDiff obj1 obj4) (isDiff obj1
obj5) (isDiff obj1 obj6) (isDiff obj1 obj7) (isDiff obj1 obj8)
(isDiff obj1 obj9) (isDiff obj1 obj0)

(isDiff obj2 obj1) (isDiff obj2 obj3) (isDiff obj2 obj4) (isDiff obj2
obj5) (isDiff obj2 obj6) (isDiff obj2 obj7) (isDiff obj2 obj8)
(isDiff obj2 obj9) (isDiff obj2 obj0)

(isDiff obj3 obj2) (isDiff obj3 obj1) (isDiff obj3 obj4) (isDiff obj3
obj5) (isDiff obj3 obj6) (isDiff obj3 obj7) (isDiff obj3 obj8)
(isDiff obj3 obj9) (isDiff obj3 obj0)

(isDiff obj4 obj2) (isDiff obj4 obj3) (isDiff obj4 obj1) (isDiff obj4
obj5) (isDiff obj4 obj6) (isDiff obj4 obj7) (isDiff obj4 obj8)
(isDiff obj4 obj9) (isDiff obj4 obj0)

(isDiff obj5 obj2) (isDiff obj5 obj3) (isDiff obj5 obj4) (isDiff obj5
obj1) (isDiff obj5 obj6) (isDiff obj5 obj7) (isDiff obj5 obj8)
(isDiff obj5 obj9) (isDiff obj5 obj0)

(isDiff obj6 obj2) (isDiff obj6 obj3) (isDiff obj6 obj4) (isDiff obj6
obj5) (isDiff obj6 obj1) (isDiff obj6 obj7) (isDiff obj6 obj8)
(isDiff obj6 obj9) (isDiff obj6 obj0)

(isDiff obj7 obj2) (isDiff obj7 obj3) (isDiff obj7 obj4) (isDiff obj7
```

```
obj5) (isDiff obj7 obj6) (isDiff obj7 obj1) (isDiff obj7 obj8)
(isDiff obj7 obj9) (isDiff obj7 obj0)

   (isDiff obj8 obj2) (isDiff obj8 obj3) (isDiff obj8 obj4) (isDiff obj8
obj5) (isDiff obj8 obj6) (isDiff obj8 obj7) (isDiff obj8 obj1)
(isDiff obj8 obj9) (isDiff obj8 obj0)

   (isDiff obj9 obj2) (isDiff obj9 obj3) (isDiff obj9 obj4) (isDiff obj9
obj5) (isDiff obj9 obj6) (isDiff obj9 obj7) (isDiff obj9 obj8)
(isDiff obj9 obj1) (isDiff obj9 obj0)

   (isDiff obj0 obj2) (isDiff obj0 obj3) (isDiff obj0 obj4) (isDiff obj0
obj5) (isDiff obj0 obj6) (isDiff obj0 obj7) (isDiff obj0 obj8)
(isDiff obj0 obj9) (isDiff obj0 obj1)
)
   (:goal (and (isFullCooked eggs) (in eggs plate0) (in salt eggs)
(isSmoothie strawberry) (on icecream strawberry) (in strawberry glass) )
)
)
```

APPENDIX C

CODEBASE AND DATA

The software packages written and used throughout this thesis can be found on Github: https://github.com/Lnair1993/Tool_Macgyvering

The packages are listed below with a high-level description of the package. Each package has its own README or documentation can be found in the specific documentation repository on Github.

The code repository has the following six folders, each with its own README:

- **tool-construction-taskPlanner**: This folder contains code for performing task planning for tasks that involve tool construction. The general idea here is to compute some score (any good measure) that indicates how good object combinations are for constructing tools, and use that score within the planner.

- **visual-score-prediction**: This folder contains code that takes pre-trained models to predict and save the scores in csv files that are used by the task planner above. This folder does not contain code for training the models. Instead, for generating csv files from trained models.

- **Dataset**: This folder contains 58 point clouds corresponding to the objects that we tested in our work. It also has an image of the complete dataset along with a table of object labels and images. The physical dataset is not currently available and we simply collected objects that were present in our lab. More standard benchmarks and datasets are yet to be developed.

- **Feature-extraction**: This folder contains the code for extracting ESF features using PCL and C++. The SCiO spectrometer directly outputs spectral readings, and the spectral readings can be processed using the code available in visual-score-prediction.

- **Training-models**: This folder contains code for training models for performing shape, material and attachment predictions. These models can then be saved as joblib files, and used by the functions in visual-score-prediction folder.

- **Auxiliary**: The code in this folder include additional models that were developed during the course of this research. This includes dual neural networks that were built for shape and material matching.

REFERENCES

[1] O. E. Dictionary and E. Idioms, *Oxford references online*, 1989.

[2] L. Zlotoff, "Macgyver television series," *USA: Paramount Pictures*, 1985.

[3] S. Cass, "Apollo 13, we have a solution," *IEEE Spectrum On-line, 04*, vol. 1, 2005.

[4] M. Turner, L. Duggan, B. Glezerson, and S. Marshall, "Thinking outside the (acrylic) box: A framework for the local use of custom-made medical devices," *Anaesthesia*, 2020.

[5] A. Maia Chagas, J. C. Molloy, L. L. Prieto-Godino, and T. Baden, "Leveraging open hardware to alleviate the burden of covid-19 on global health systems," *Plos Biology*, vol. 18, no. 4, e3000730, 2020.

[6] O. E. D. Kitchen, *Apollobvm - emergency use ventilator*, http://oedk.rice.edu/apollobvm/, 2020.

[7] T. B. Jones and A. C. Kamil, "Tool-making and tool-using in the northern blue jay," *Science*, vol. 180, no. 4090, pp. 1076–1078, 1973.

[8] T. Hashmi, *Baby wipes mousepad*, https://www.flickr.com/photos/61697474@N00/4471742345, 2010.

[9] M. Schoeler and F. Wörgötter, "Bootstrapping the semantics of tools: Affordance analysis of real world objects on a per-part basis," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, no. 2, pp. 84–98, 2015.

[10] P. Abelha, F. Guerin, and M. Schoeler, "A model-based approach to finding substitute tools in 3d vision data," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 2471–2478.

[11] P. Abelha and F. Guerin, "Learning how a tool affords by simulating 3d models from the web," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 4923–4929.

[12] A. Boteanu, D. Kent, A. Mohseni-Kabir, C. Rich, and S. Chernova, "Towards robot adaptability in new situations," in *2015 AAAI Fall Symposium Series, Arlington, VA: AAAI Press*, 2015.

[13] A. Agostini, M. J. Aein, S. Szedmak, E. E. Aksoy, J. Piater, and F. Würgütter, "Using structural bootstrapping for object substitution in robotic executions of human-like

manipulation tasks," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 6479–6486.

[14] N. Cutting, I. A. Apperly, and S. R. Beck, "Why do children lack the flexibility to innovate tools?" *Journal of experimental child psychology*, vol. 109, no. 4, pp. 497–511, 2011.

[15] N. Cutting, I. A. Apperly, J. Chappell, and S. R. Beck, "Is tool modification more difficult than innovation?" *Cognitive development*, vol. 52, p. 100 811, 2019.

[16] S. R. Beck, N. Cutting, I. A. Apperly, Z. Demery, L. Iliffe, S. Rishi, and J. Chappell, "Is tool-making knowledge robust over time and across problems?" *Frontiers in psychology*, vol. 5, p. 1395, 2014.

[17] S. R. Beck, I. A. Apperly, J. Chappell, C. Guthrie, and N. Cutting, "Making tools isn't child's play," *Cognition*, vol. 119, no. 2, pp. 301–306, 2011.

[18] M. O. Ernst and H. H. Bülthoff, "Merging the senses into a robust percept," *Trends in cognitive sciences*, vol. 8, no. 4, pp. 162–169, 2004.

[19] J. Lupo and M. Barnett-Cowan, "Perceived object stability depends on shape and material properties," *Vision research*, vol. 109, pp. 158–165, 2015.

[20] L. Nair, J. Balloch, and S. Chernova, "Tool macgyvering: Tool construction using geometric reasoning," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 5837–5843.

[21] L. Nair, N. Srikanth, Z. Erikson, and S. Chernova, "Autonomous tool construction using part shape and attachment prediction," in *Proceedings of Robotics: Science and Systems*, 2019, pp. 1–10.

[22] L. Nair, N. Shrivatsav, and S. Chernova, "Tool macgyvering: A novel framework for combining tool substitution and construction," *IEEE Transactions on Robotics (T-RO). Under review*, 2020.

[23] N. Shrivatsav, L. Nair, and S. Chernova, "Tool substitution with shape and material reasoning using dual neural networks," *arXiv preprint arXiv:1911.04521*, 2019.

[24] L. Velayudhan and R. C. Arkin, "Sloth and slow loris inspired behavioral controller for a robotic agent," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2017, pp. 1880–1885.

[25] L. Nair and S. Chernova, "Feature guided search for creative problem solving through tool construction," in *Frontiers in Robotics and AI*, 2020.

[26] D. S. Weld, "Recent advances in ai planning," *AI magazine*, vol. 20, no. 2, pp. 93–93, 1999.

[27] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, *Pddl-the planning domain definition language*, 1998.

[28] V. Sarathy and M. Scheutz, "Macgyver problems: Ai challenges for testing resourcefulness and creativity," *Advances in Cognitive Systems*, vol. 6, pp. 31–44, 2018.

[29] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Engineering*, vol. 5, no. 1, p. 1 502 242, 2018.

[30] K. Deb, "Multi-objective optimization," in *Search methodologies*, Springer, 2014, pp. 403–449.

[31] V. Sarathy and M. Scheutz, "The macgyver test-a framework for evaluating machine resourcefulness and creative problem solving," *arXiv preprint arXiv:1704.08350*, 2017.

[32] E. Gizzi, L. Nair, J. Sinapov, and S. Chernova, "From computational creativity to creative problem solving agents," *International Conference on Computational Creativity (ICCC)*, 2020.

[33] V. Sarathy, "Real world problem-solving," *Frontiers in human neuroscience*, vol. 12, p. 261, 2018.

[34] A.-M. Olteţeanu and Z. Falomir, "Object replacement and object composition in a creative cognitive system. towards a computational solver of the alternative uses test," *Cognitive Systems Research*, vol. 39, pp. 15–32, 2016.

[35] R. Freedman, S. Friedman, D. Musliner, and M. Pelican, "Creative problem solving through automated planning and analogy," in *AAAI 2020 Workshop on Generalization in Planning (GenPlan 20)*, 2020.

[36] E. Gizzi, M. G. Castro, and J. Sinapov, "Creative problem solving by robots using action primitive discovery," in *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, IEEE, 2019, pp. 228–233.

[37] A. Suárez-Hernández, J. Segovia-Aguas, C. Torras, and G. Alenyà, "Strips action discovery," *arXiv preprint arXiv:2001.11457*, 2020.

[38] J. Sinapov and A. Stoytchev, "Learning and generalization of behavior-grounded tool affordances," in *2007 IEEE 6th International Conference on Development and Learning*, IEEE, 2007, pp. 19–24.

[39] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *2008 7th IEEE International Conference on Development and Learning*, IEEE, 2008, pp. 91–96.

[40] S. Brown and C. Sammut, "A relational approach to tool-use learning in robots," in *International Conference on Inductive Logic Programming*, Springer, 2012, pp. 1–15.

[41] N. Yamanobe, W. Wan, I. G. Ramirez-Alpizar, D. Petit, T. Tsuji, S. Akizuki, M. Hashimoto, K. Nagata, and K. Harada, "A brief review of affordance in robotic manipulation research," *Advanced Robotics*, vol. 31, no. 19-20, pp. 1086–1101, 2017.

[42] C. Erdogan and M. Stilman, "Planning in constraint space: Automated design of functional structures," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 1807–1812.

[43] T. Tosun, J. Daudelin, G. Jing, H. Kress-Gazit, M. Campbell, and M. Yim, "Perception-informed autonomous environment augmentation with modular robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6818–6824.

[44] M. Saboia, V. Thangavelu, W. Gosrich, and N. Napp, "Autonomous adaptive modification of unstructured environments," *Robotics: Science and Systems (RSS 2018)*, 2018.

[45] D. Choi, P. Langley, and S. T. To, "Creating and using tools in a hybrid cognitive architecture," in *2018 AAAI Spring Symposium Series*, 2018.

[46] M. Stilman, M. Zafar, C. Erdogan, P. Hou, S. Reynolds-Haertle, and G. Tracy, "Robots using environment objects as tools the 'macgyver' paradigm for mobile manipulation," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 2568–2568.

[47] M. Levihn and M. Stilman, "Using environment objects as tools: Unconventional door opening," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, 2014, pp. 2502–2508.

[48] C. Erdogan and M. Stilman, "Autonomous realization of simple machines," in *Experimental Robotics*, Springer, 2016, pp. 471–486.

[49] H. Wicaksono and C. S. R. Sheh, "Towards explainable tool creation by a robot," in *IJCAI-17 Workshop on Explainable AI (XAI)*, 2017, p. 63.

[50] L. Wang, L. Brodbeck, and F. Iida, "Mechanics and energetics in tool manufacture and use: A synthetic approach," *Journal of the Royal Society Interface*, vol. 11, no. 100, p. 20 140 827, 2014.

[51] M. Schoeler and F. Wörgötter, "Bootstrapping the semantics of tools: Affordance analysis of real world objects on a per-part basis," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, no. 2, pp. 84–98, 2016.

[52] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3d object classification," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, IEEE, 2011, pp. 2987–2992.

[53] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012.

[54] A. H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer graphics and Applications*, vol. 1, no. 1, pp. 11–23, 1981.

[55] J. Perlow, B. Rosman, B. Hayes, and P. Ranchod, "Raw material selection for object construction," in *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, IEEE, 2017, pp. 144–149.

[56] G. Schwartz and K. Nishino, "Recognizing material properties from images," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[57] D. Hu, L. Bo, and X. Ren, "Toward robust material recognition for everyday objects.," in *BMVC*, Citeseer, vol. 2, 2011, p. 6.

[58] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3479–3487.

[59] Z. Erickson, N. Luskey, S. Chernova, and C. Kemp, "Classification of household materials via spectroscopy," *IEEE Robotics and Automation Letters*, 2019.

[60] R. C. Arkin, M. Fujita, T. Takagi, and R. Hasegawa, "An ethological and emotional basis for human–robot interaction," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 191–201, 2003.

[61] M. Likhachev and R. C. Arkin, "Robotic comfort zones," in *Sensor Fusion and Decentralized Control in Robotic Systems III*, International Society for Optics and Photonics, vol. 4196, 2000, pp. 27–41.

[62] D. Stout and T. Chaminade, "The evolutionary neuroscience of tool making," *Neuropsychologia*, vol. 45, no. 5, pp. 1091–1100, 2007.

[63] D. Stout, "Stone toolmaking and the evolution of human culture and cognition," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 366, no. 1567, pp. 1050–1059, 2011.

[64] C. Boesch and H. Boesch, "Tool use and tool making in wild chimpanzees," *Folia primatologica*, vol. 54, no. 1-2, pp. 86–99, 1990.

[65] N. Toth, K. D. Schick, E. S. Savage-Rumbaugh, R. A. Sevcik, and D. M. Rumbaugh, "Pan the tool-maker: Investigations into the stone tool-making and tool-using capabilities of a bonobo (pan paniscus)," *Journal of Archaeological Science*, vol. 20, no. 1, pp. 81–91, 1993.

[66] C. D. Bird and N. J. Emery, "Insightful problem solving and creative tool modification by captive nontool-using rooks," *Proceedings of the National Academy of Sciences*, vol. 106, no. 25, pp. 10 370–10 375, 2009.

[67] N. Lang, C. Hudgings, A. Jacox, J. Lancour, M. McClure, K. McCormick, V. Saba, T. Stenvig, R. Zielstorff, P. Prescott, *et al.*, "Toward a national database for nursing practice," *NM Lang (Ed.), Nursing Data Systems: An Emerging Framework. Washington, DC: American Nurses Publishing*, 1995.

[68] J. J. Gibson, "Perceiving, acting, and knowing: Toward an ecological psychology," *The Theory of Affordances*, pp. 67–82, 1977.

[69] E. Şahin, M. Çakmak, M. R. Doğar, E. Uğur, and G. Üçoluk, "To afford or not to afford: A new formalization of affordances toward affordance-based robot control," *Adaptive Behavior*, vol. 15, no. 4, pp. 447–472, 2007.

[70] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory–motor coordination to imitation," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 15–26, 2008.

[71] M. Lopes, F. S. Melo, and L. Montesano, "Affordance-based imitation learning in robots," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, IEEE, 2007, pp. 1015–1021.

[72] M. Andries, R. O. Chavez-Garcia, R. Chatila, A. Giusti, and L. M. Gambardella, "Affordance equivalences in robotics: A formalism," *Frontiers in Neurorobotics*, vol. 12, p. 26, 2018.

[73] Z. Erickson, E. Xing, B. Srirangam, S. Chernova, and C. C. Kemp, "Multimodal material classification for robots using spectroscopy and high resolution texture imaging," *arXiv preprint arXiv:2004.01160*, 2020.

[74] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features.," in *International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1374–1381.

[75] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Conference on Robot Learning*, 2017, pp. 109–118.

[76] T. Fitzgerald, A. K. Goel, and A. L. Thomaz, "Representing skill demonstrations for adaptation and transfer," in *AAAI Symposium on Knowledge, Skill, and Behavior Transfer in Autonomous Robots*, 2014.

[77] P. Gajewski, P. Ferreira, G. Bartels, C. Wang, F. Guerin, B. Indurkhya, M. Beetz, and B. Śniezyński, "Adapting everyday manipulation skills to varied scenarios," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 1345–1351.

[78] Y. Shiraki, K. Nagata, N. Yamanobe, A. Nakamura, K. Harada, D. Sato, and D. N. Nenchev, "Modeling of everyday objects for semantic grasp," in *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, IEEE, 2014, pp. 750–755.

[79] D. Beßler, M. Pomarlan, and M. Beetz, "Owl-enabled assembly planning for robotic agents," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1684–1692.

[80] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[81] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[82] P. Zech and J. Piater, "Grasp learning by sampling from demonstration," *arXiv preprint arXiv:1611.06366*, 2016.

[83] R. Tanese, "Distributed genetic algorithms for function optimization," 1989.

[84] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[85] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[86] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[87] M. Tamosiunaite, M. J. Aein, J. M. Braun, T. Kulvicius, I. Markievicz, J. Kapociute-Dzikiene, R. Valteryte, A. Haidu, D. Chrysostomou, B. Ridge, *et al.*, "Cut & recombine: Reuse of robot action components based on simple language instructions," *The International Journal of Robotics Research*, p. 0 278 364 919 865 594, 2019.

[88] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, 2015.

[89] T.-T. Do, A. Nguyen, and I. Reid, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 1–5.

[90] Y. Zhu, A. Fathi, and L. Fei-Fei, "Reasoning about object affordances in a knowledge base representation," in *European conference on computer vision*, Springer, 2014, pp. 408–424.

[91] E. Karpas and C. Domshlak, "Cost-optimal planning with landmarks.," in *IJCAI*, 2009, pp. 1728–1733.

[92] J. Hoffmann and B. Nebel, "The ff planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[93] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[94] Z. Zhang, N. R. Sturtevant, R. C. Holte, J. Schaeffer, and A. Felner, "A* search with inconsistent heuristics.," in *IJCAI*, 2009, pp. 634–639.

[95] A. Bagchi and A. Mahanti, "Search algorithms under different kinds of heuristics—a comparative study," *Journal of the ACM (JACM)*, vol. 30, no. 1, pp. 1–21, 1983.

[96] A. Pierson and M. Schwager, "Adaptive inter-robot trust for robust multi-robot sensor coverage," in *Robotics Research*, Springer, 2016, pp. 167–183.

[97] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial intelligence*, vol. 1, no. 3-4, pp. 193–204, 1970.